GaTT – Software Design Specification

Build Version: 3 built on $Date: 2005−11−11 01:22:10 −0500 (Fri, 11 Nov 2005) $

Prepared for BC/BSM by TeamBC

Table of Contents

1 Introduction **1**
1.1 Goals and objectives 1
1.2 Statement of scope 1
1.3 Software context 1
1.4 Expected software response 2

2 Data Design **3**
2.1 Data Structures 3
2.2 Database description 3

3 Architectural Design **8**
3.1 Program Structure 8
3.1.1 Architectural Diagram 8

4 Component−Level Design **9**
4.1 Description of Add Group 9
4.1.1 Processing Narrative 9
4.1.2 Interface Description 9
4.1.3 Component Processing Detail 9
4.2 Description of Add User 12
4.2.1 Processing Narrative 12
4.2.2 Interface Description 12
4.2.3 Component Processing Detail 12
4.3 Description of View Group List 13
4.3.1 Processing Narrative 13
4.3.2 Interface Description 14
4.3.3 Component Processing Detail 14
4.4 Description of View User List 14
4.4.1 Processing Narrative 14
4.4.2 Interface Description 14
4.4.3 Component Processing Detail 15
4.5 Description of View Group 15
4.5.1 Processing Narrative 15
4.5.2 Interface Description 15
4.5.3 Component Processing Detail 16
4.6 Description of View User 18
4.6.1 Processing Narrative 18
4.6.2 Interface Description 18
4.6.3 Component Processing Detail 18
4.7 Description of Modify Group 19
4.7.1 Processing Narrative 19
4.7.2 Interface Description 19
4.7.3 Component Processing Detail 20
4.8 Description of Modify User 24
4.8.1 Processing Narrative 25
4.8.2 Interface Description 25
4.8.3 Component Processing Detail 25
4.9 Description of Delete Group 26

Table of Contents

4 Component–Level Design

4.9.1 Processing Narrative....26
4.9.2 Interface Description....26
4.9.3 Component Processing Detail....26
4.10 Description of Delete User....27
4.10.1 Processing Narrative....27
4.10.2 Interface Description....28
4.10.3 Component Processing Detail....28
4.11 Description of Log In....28
4.11.1 Processing Narrative....29
4.11.2 Interface Description....29
4.11.3 Component Processing Detail....29
4.12 Description of Change Password....30
4.12.1 Processing Narrative....30
4.12.2 Interface Description....30
4.12.3 Component Processing Detail....30
4.13 Description of Log Out....31
4.13.1 Processing Narrative....31
4.13.2 Interface Description....31
4.13.3 Component Processing Detail....32
4.14 Description of Internal Link to an Object....33
4.14.1 Processing Narrative....33
4.14.2 Interface Description....34
4.14.3 Component Processing Detail....34
4.15 Description of External Link to an Object....35
4.15.1 Processing Narrative....36
4.15.2 Interface Description....36
4.15.3 Component Processing Detail....36
4.16 Description of Add Attachement to an Item....37
4.16.1 Processing Narrative....38
4.16.2 Interface Description....38
4.16.3 Component Processing Detail....38
4.17 Description of Change Log for Watch Item....39
4.17.1 Processing Narrative....40
4.17.2 Interface Description....40
4.17.3 Component Processing Detail....40
4.18 Description of Change Log for Inventory Item....42
4.18.1 Processing Narrative....42
4.18.2 Interface Description....42
4.18.3 Component Processing Detail....42
4.19 Description of Email to the Assigned To Person of Watch List Item....45
4.19.1 Processing Narrative....45
4.19.2 Interface Description....45
4.19.3 Component Processing Detail....45
4.20 Description of Email to the Assigned To Person of Inventory Item....47
4.20.1 Processing Narrative....47
4.20.2 Interface Description....47
4.20.3 Component Processing Detail....47

Table of Contents

4 Component–Level Design
4.21 Description of View Inventory List....49
4.21.1 Processing Narrative....49
4.21.2 Interface Description....49
4.21.3 Component Processing Detail....49
4.22 Description of View Watch List....50
4.22.1 Processing Narrative....51
4.22.2 Interface Description....51
4.22.3 Component Processing Detail....51
4.23 Description of View Watch Object....52
4.23.1 Processing Narrative....53
4.23.2 Interface Description....53
4.23.3 Component Processing Detail....53
4.24 Description of View Inventory Object....54
4.24.1 Processing Narrative....55
4.24.2 Interface Description....55
4.24.3 Component Processing Detail....55
4.25 Description of Add Watch Object....56
4.25.1 Processing Narrative....57
4.25.2 Interface Description....57
4.25.3 Component Processing Detail....57
4.26 Description of Add Inventory Object....58
4.26.1 Processing Narrative....58
4.26.2 Interface Description....59
4.26.3 Component Processing Detail....59
4.27 Description of Modify Watch Object....60
4.27.1 Processing Narrative....60
4.27.2 Interface Description....61
4.27.3 Component Processing Detail....61
4.28 Description of Modify Inventory Object....63
4.28.1 Processing Narrative....63
4.28.2 Interface Description....63
4.28.3 Component Processing Detail....63
4.29 Description of Delete Watch Object....66
4.29.1 Processing Narrative....66
4.29.2 Interface Description....66
4.29.3 Component Processing Detail....66
4.30 Description of Delete Inventory Object....67
4.30.1 Processing Narrative....67
4.30.2 Interface Description....67
4.30.3 Component Processing Detail....67
4.31 Description of Customized Report....68
4.31.1 Processing Narrative....69
4.31.2 Interface Description....69
4.31.3 Component Processing Detail....69
4.32 Description of View Preconfigured Report....69
4.32.1 Processing Narrative....69
4.32.2 Interface Description....70

Table of Contents

4 Component–Level Design
4.32.3 Component Processing Detail....70

5 Sequence Diagrams....71

6 User interface Design....97
6.1 Description of the user interface....97
6.1.1 Watches....97
6.1.2 Inventory....97
6.1.3 Reports....98
6.1.4 User Management....99
6.1.5 Configuration....100
6.1.6 Requirements....101
6.1.7 Generic Add Window....102
6.2 Objects and actions....103
6.2.1 Watches....103
6.2.2 Inventory....103
6.2.3 Reports....103
6.2.4 User Management....103
6.2.5 Configuration....104
6.2.6 Requirements....104
6.2.7 Generic Add Window....104
6.3 Interface design rules....105
6.4 UIDS description....105

7 Restrictions, limitations, and constraints....106

8 Testing Process....107
8.1 Classes of tests....107
8.2 Performance bounds....107
8.3 Identification of critical components....107

9 Appendices....109
9.1 Requirements traceability matrix....109
9.2 Packaging and installation issues....111
9.3 Design metrics to be used....112
9.4 Supplementary information....113

10 Project Glossary....114
10.1 Core Modules....114
10.2 The Graphical User Interface....114
10.3 The Migration Tool Manual....114
10.4 Object Linking and Embedding....115
10.5 Paper Prototyping....115
10.6 Add Privilege....115
10.7 Delete Privilege....115
10.8 Modify Privilege....115
10.9 View Privilege....115

Table of Contents

10 Project Glossary
10.10 Project Inventory....116
10.11 Project Requirements....116
10.12 Project Watches....116
10.13 Project Velocity....116
10.14 Reporting Module....116
10.15 User Stories....117
10.16 The Customer....117
10.16.1 Blue Cross/Blue Shield of Michigan....117
10.16.2 Primary Customer Contact....117
10.17 The Team....117
10.18 User Group....118

11 Changelog....119

12....120
12.1 Changes for Build 3:....120
12.1.1 Changes for page: DataDesign....120
12.1.2 Changes for page: ComponentLevelDesign....120
12.1.3 Changes for page: SequenceDiagrams....120
12.1.4 Changes for page: UserInterfaceDesign....120
12.1.5 Changes for page: TestingIssues....120
12.1.6 Changes for page: AppendicesSection....120
12.1.7 Changes to documentation repository items:....120
12.2 Changes for Build 2:....121
12.2.1 Changes to documentation repository items:....121
12.3 Changes for Build 1:....122
12.3.1 Changes for page: ProjectOverviewAndScope....122
12.3.2 Changes for page: DataDesign....122
12.3.3 Changes for page: ComponentLevelDesign....123
12.3.4 Changes for page: SequenceDiagrams....123
12.3.5 Changes for page: UserInterfaceDesign....124
12.3.6 Changes for page: RestrictionsLimitations....124
12.3.7 Changes for page: TestingIssues....124
12.3.8 Changes for page: AppendicesSection....124
12.3.9 Changes for page: ProjectGlossary....124
12.3.10 Changes for page: CoreModules....125
12.3.11 Changes for page: GraphicalUserInterface....125
12.3.12 Changes for page: MigrationToolManual....125
12.3.13 Changes for page: ObjectLinkingAndEmbedding....125
12.3.14 Changes for page: PaperPrototyping....125
12.3.15 Changes for page: AddPrivilege....125
12.3.16 Changes for page: DeletePrivilege....126
12.3.17 Changes for page: ModifyPrivilege....126
12.3.18 Changes for page: ViewPrivilege....126
12.3.19 Changes for page: ProjectInventory....126
12.3.20 Changes for page: ProjectRequirements....126
12.3.21 Changes for page: ProjectWatches....126

Table of Contents

12

12.3.22 Changes for page: ProjectVelocity....127
12.3.23 Changes for page: ReportingModule....127
12.3.24 Changes for page: UserStories....127
12.3.25 Changes for page: TheCustomer....127

1 Introduction

The key to effective management is the communication of project requirements, goals, and risks to the project team. The NPI team at BCBSM has developed a tool for this precise purpose. The tool tracks requirements, designs, and issues (such as risks). The team can track the progress on the project, and issues which still need to be addressed.

The tracking tool developed by the NPI project team is an asset to technical management of projects. BCBSM would like to develop a generic tracking tool similar to the NPI team's tool. This tool would perform the same functions as the specialized tool, but would be general enough to be used by other large projects at BCBSM. Team BC paid careful attention to the needs of BCBSM through a personal interview with the NPI team and we came up with an effective application to meet those needs. The following pages explain and give more detailed information about the requirements and the specifications of this project.

1.1 Goals and objectives

Using the tracking tool developed by the NPI team as a model, Team BC is to develop an enhanced generic tracking tool that may be used by large technical projects at BCBSM.

1.2 Statement of scope

The Tracking tool will be architected to support the functional requirements listed below. Each requirement will be described in detail in the Functional Requirements section.

- Tool must track information
  - Issues
  - Inventory
  - Requirements
- Tool must create references between items in and out of the tracking modules
- Tool must provide the following privileges to a group based access security system:
  - View Privilege
  - Create Privilege
  - Update Privilege
  - Configure Privilege
  - Delete Privilege
- All core modules should be configurable

1.3 Software context

The goal of the software is to improve the communication of project requirements, risks, issues, and goals and enhance collaborative development within a software project team.

1.4 Expected software response

The software should response as follows:

1. Information about an object should be added to the database if the user(with proper access privileges) wants to add an object to the tool.
2. Information should be updated in the database if the user(with proper access privileges) wants to modify an object in the tool.
3. Information should be deleted from the database if the user(with proper access privileges) wants to remove an object from the tool.
4. Users should only be able to access the tool if they have a valid username and password.
5. User should only be able to access the modules of the tool for which they have privileges.

2 Data Design

2.1 Data Structures

Since the development platform, Microsoft Access, is essentially a prebuilt interface to the database, there are actually very few data structures, internal or external, and most of those are defined by the Access platform itself. Because the platform defines almost all of the data structures we will be using, there is little need to detail them here. Instead, refer to the Access help and Microsoft's online MSDN library. The primary library that will be used is the ActiveX Data Objects Database (ADODB) library. This library will be used for any programmatic access to the database (as opposed to queries bound to forms and reports). In addition to ADODB, we will also use portions of the Access Automation interface, which will allow us to automate some of the work of creating forms and reports. Other objects will include the various user interface widgets, which again are defined by the Access platform.

2.2 Database description

The most important part of the system is the actual database. It's design is detailed in the Entity−Relationship Diagram (ERD) below. Basically, there are three major relationship areas in the database−−object storage, users, groups and permissions, and auditing (change tracking and login tracking).

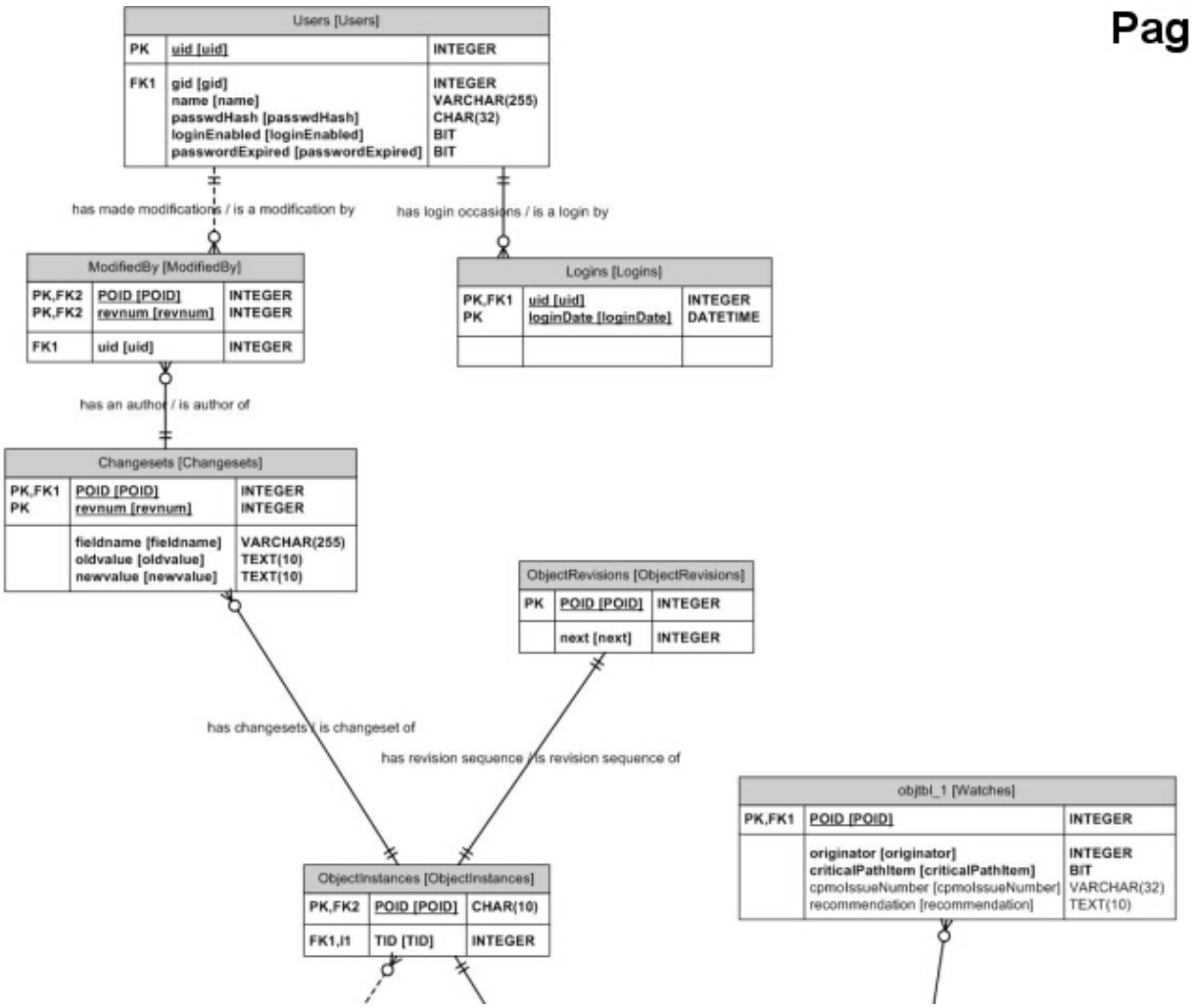
The first major area of the database is object storage. Object storage is essentially a tree of related tables. Each object has a type which has a name, an ID, and a description. This information is stored in the ObjectTypes table. Each object type corresponds to a specific type of tracked object−−Watches, Inventory, or Requirements. In addition, an object type can be derived from a parent type, and inherits all of the attributes of that parent type. For each defined object type there is a table which both defines its attributes, and stores instances of that object type. Each object stored in the database has a globally unique persistent object id (POID). This POID is used to join all of the tables which make up the instantiated object. This is basically an implementation of the Map Each Class To Its Own Table design pattern. This pattern was chosen because it greatly simplifies the process of creating new object types, a process that will occur often as each individual project configures the tool for its own use. It also simplifies the queries required to create views of the objects in the database; essentially, views can be created that join all of the tables that make up an object instance, which in turn makes it easier for end users to write queries for custom reports.

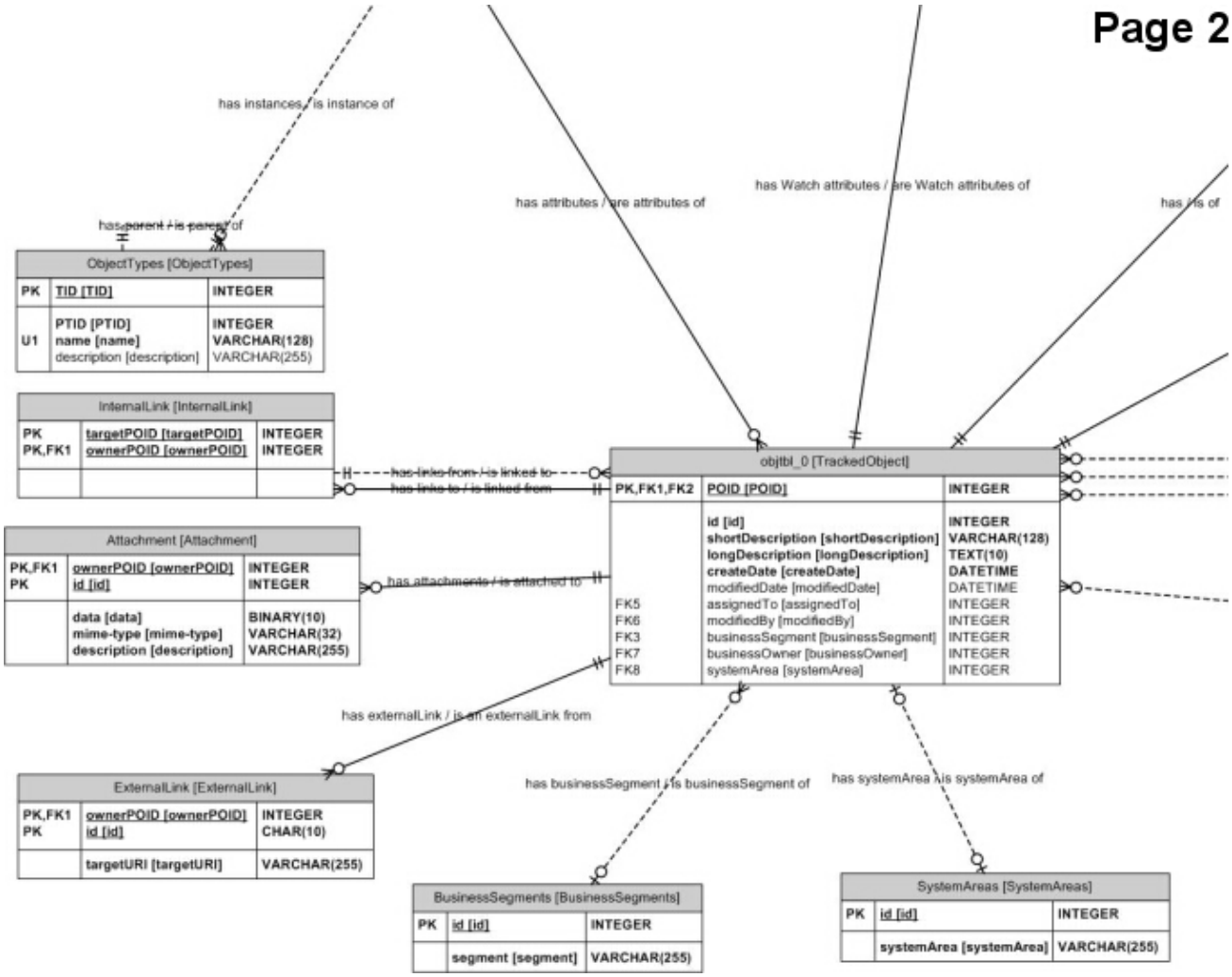
In addition to the object storage portion of the database, there is also a storage area for users, groups, and their privileges. This security portion of the database is fairly straightforward. Users consist of a unique id, a primary group, a hashed password, and two flags indicating whether or not the user is disabled, and whether the user's password has been expired. At this time there is no automated method for expiring a password; it is simply a method by which an administrator can force a user to change a password. The password itself is stored as an MD5 hash, and is therefore unrecoverable once stored. The primary group is used to determine a users privileges; no other group memberships impact privileges. Groups and users are joined using the GroupMemberships table. A group's privilege list is stored, not surprisingly, as a set of rows in the PrivilegeLists table, and the names of the actual privileges are stored in the Privileges table.

The last section of the database is concerned with auditing the users' actions within the tool. It consists of two tables−−the Changesets table and Logins table. The Logins table simply stores a timestamp and a user id for every login to the database. The Changesets table is slightly more complicated. For every change to a stored object, the field name of each field changed is stored, along with the old value and the new value (each converted to a text representation). The author of a changeset is joined to the changeset using the ModifiedBy table.

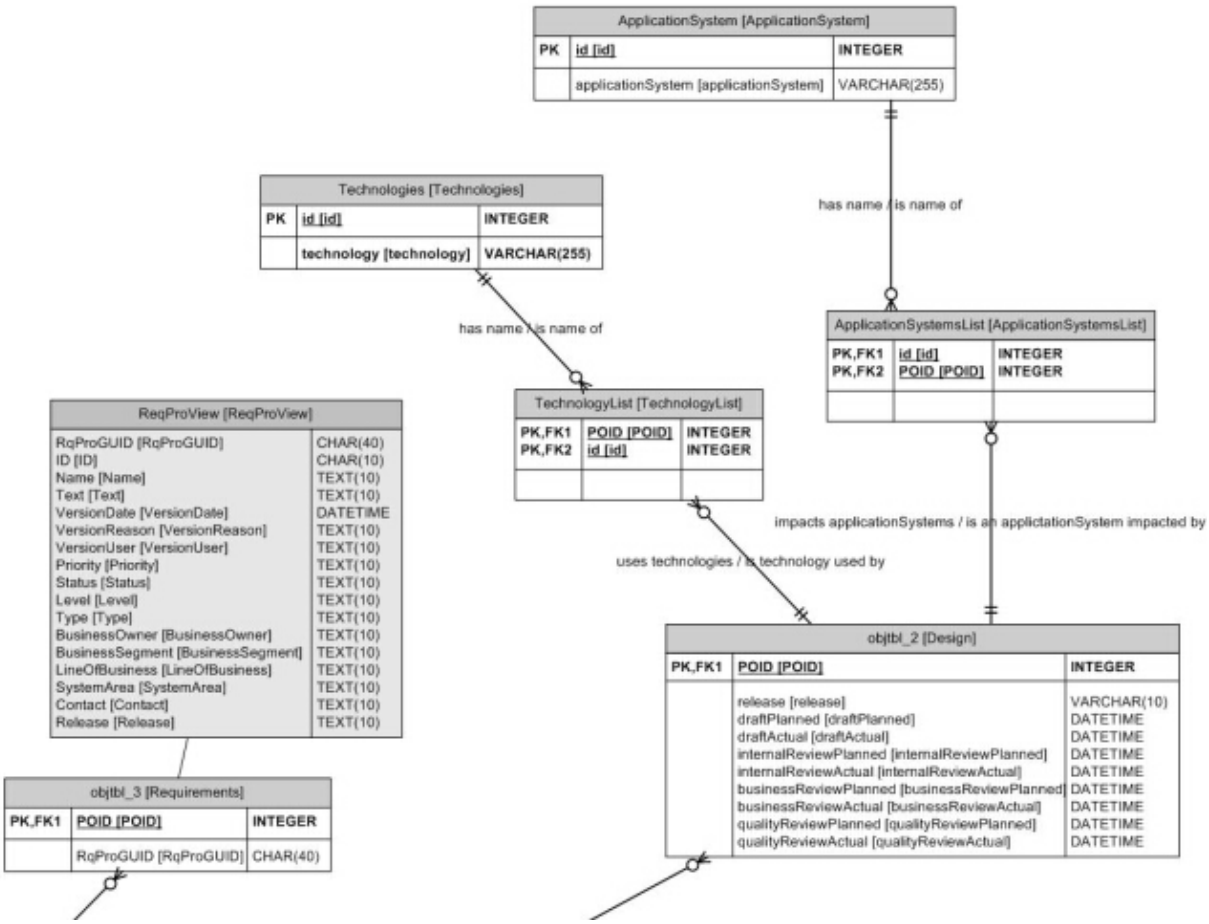
The ERD below has been split into four pages, each labeled. These four pages should be assembled in the following order:

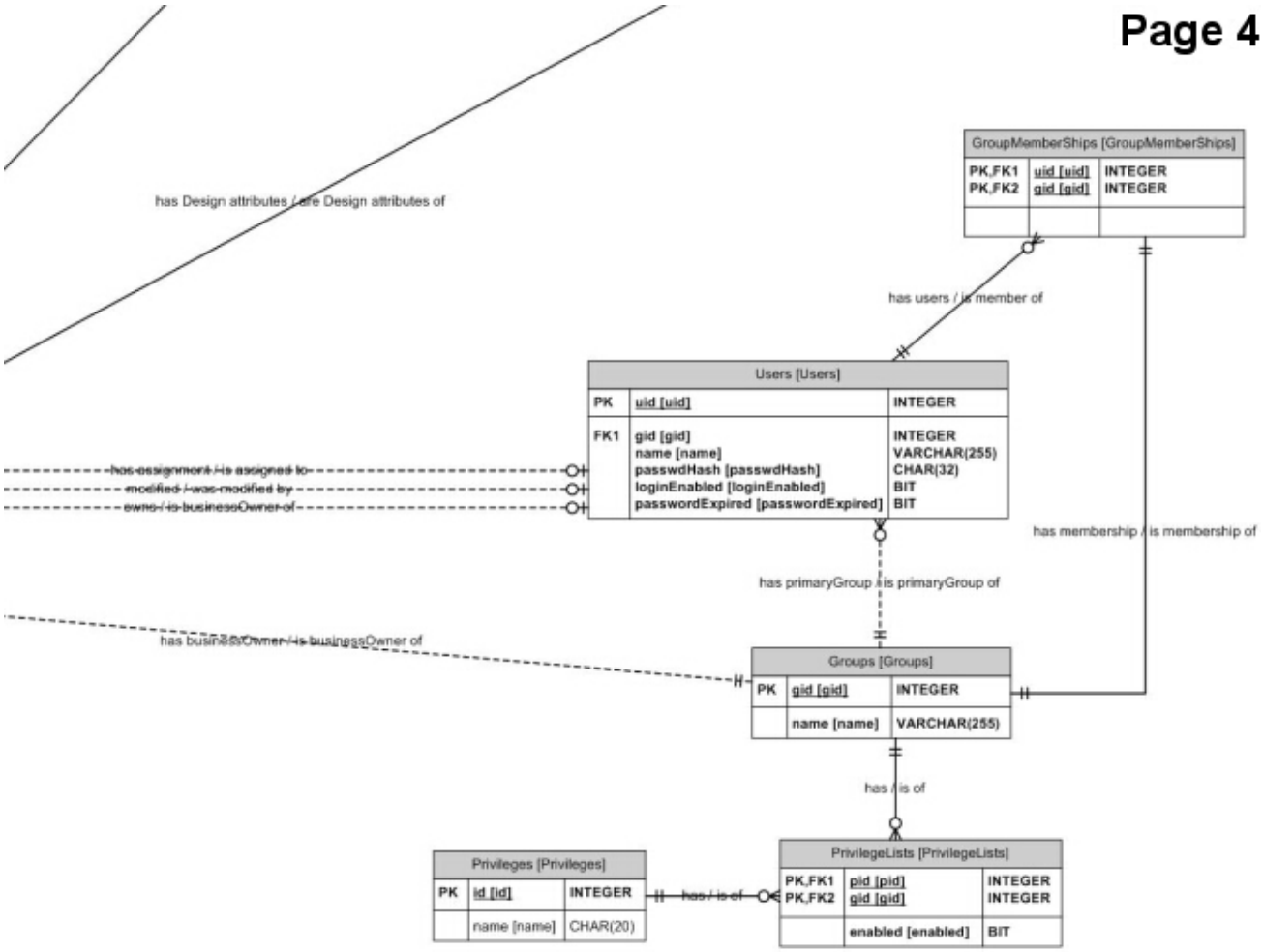
1	3
2	4




Page 2
has instances / is instance of
has attributes / are attributes of
has Watch attributes / are Watch attributes of
has / is of
has parent / is parent of
ObjectTypes [ObjectTypes]
PK
TID [TID]
INTEGER
U1
PTID [PTID]
name [name]
description [description]
INTEGER
VARCHAR(128)
VARCHAR(255)
InternalLink [InternalLink]
PK
PK,FK1
targetPOID [targetPOID]
ownerPOID [ownerPOID]
INTEGER
INTEGER
has links from / is linked to
has links to / is linked from
objtbl_0 [TrackedObject]
PK,FK1,FK2
POID [POID]
INTEGER
id [id]
shortDescription [shortDescription]
longDescription [longDescription]
createDate [createDate]
modifiedDate [modifiedDate]
assignedTo [assignedTo]
modifiedBy [modifiedBy]
businessSegment [businessSegment]
businessOwner [businessOwner]
systemArea [systemArea]
INTEGER
VARCHAR(128)
TEXT(10)
DATETIME
DATETIME
INTEGER
INTEGER
INTEGER
INTEGER
INTEGER
FK5
FK6
FK3
FK7
FK8
Attachment [Attachment]
PK,FK1
PK
ownerPOID [ownerPOID]
id [id]
INTEGER
INTEGER
data [data]
mime-type [mime-type]
description [description]
BINARY(10)
VARCHAR(32)
VARCHAR(255)
has attachments / is attached to
has externalLink / is an externalLink from
ExternalLink [ExternalLink]
PK,FK1
PK
ownerPOID [ownerPOID]
id [id]
INTEGER
CHAR(10)
targetURI [targetURI]
VARCHAR(255)
has businessSegment / is businessSegment of
has systemArea / is systemArea of
BusinessSegments [BusinessSegments]
PK
id [id]
INTEGER
segment [segment]
VARCHAR(255)
SystemAreas [SystemAreas]
PK
id [id]
INTEGER
systemArea [systemArea]
VARCHAR(255)

Page 3


ApplicationSystem [ApplicationSystem]
PK
id [id]
INTEGER
applicationSystem [applicationSystem]
VARCHAR(255)
has name / is name of
Technologies [Technologies]
PK
id [id]
INTEGER
technology [technology]
VARCHAR(255)
has name / is name of
ApplicationSystemsList [ApplicationSystemsList]
PK,FK1
id [id]
INTEGER
PK,FK2
POID [POID]
INTEGER
TechnologyList [TechnologyList]
PK,FK1
POID [POID]
INTEGER
PK,FK2
id [id]
INTEGER
impacts applicationSystems / is an applictationSystem impacted by
uses technologies / is technology used by
ReqProView [ReqProView]
RqProGUID [RqProGUID] CHAR(40)
ID [ID] CHAR(10)
Name [Name] TEXT(10)
Text [Text] TEXT(10)
VersionDate [VersionDate] DATETIME
VersionReason [VersionReason] TEXT(10)
VersionUser [VersionUser] TEXT(10)
Priority [Priority] TEXT(10)
Status [Status] TEXT(10)
Level [Level] TEXT(10)
Type [Type] TEXT(10)
BusinessOwner [BusinessOwner] TEXT(10)
BusinessSegment [BusinessSegment] TEXT(10)
LineOfBusiness [LineOfBusiness] TEXT(10)
SystemArea [SystemArea] TEXT(10)
Contact [Contact] TEXT(10)
Release [Release] TEXT(10)
objtbl_2 [Design]
PK,FK1
POID [POID]
INTEGER
release [release] VARCHAR(10)
draftPlanned [draftPlanned] DATETIME
draftActual [draftActual] DATETIME
internalReviewPlanned [internalReviewPlanned] DATETIME
internalReviewActual [internalReviewActual] DATETIME
businessReviewPlanned [businessReviewPlanned] DATETIME
businessReviewActual [businessReviewActual] DATETIME
qualityReviewPlanned [qualityReviewPlanned] DATETIME
qualityReviewActual [qualityReviewActual] DATETIME
objtbl_3 [Requirements]
PK,FK1
POID [POID]
INTEGER
RqProGUID [RqProGUID]
CHAR(40)

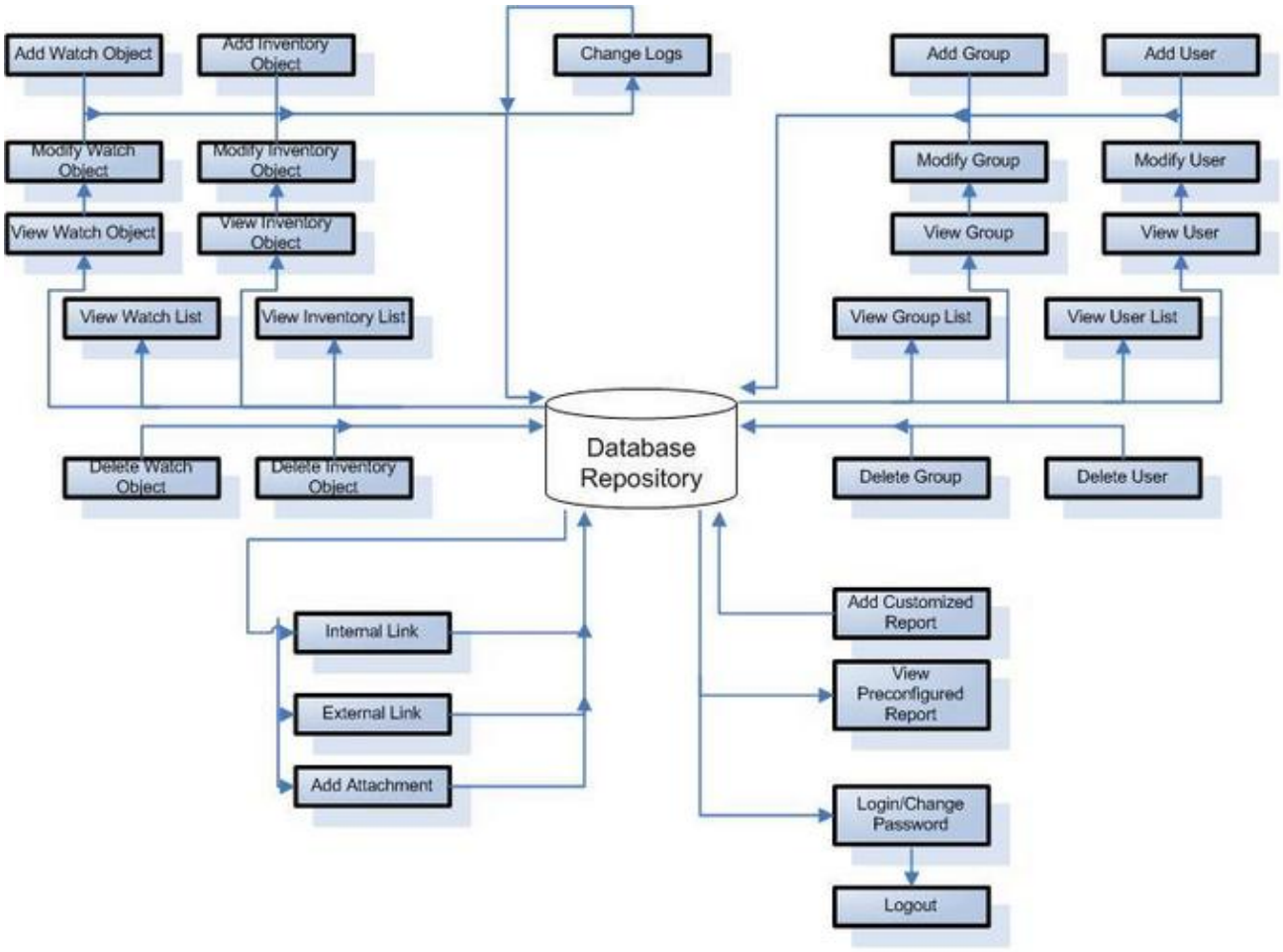

Page 4
GroupMemberShips [GroupMemberShips]
PK,FK1 uid [uid] INTEGER
PK,FK2 gid [gid] INTEGER
has Design attributes / are Design attributes of
has users / is member of
Users [Users]
PK uid [uid] INTEGER
FK1 gid [gid] INTEGER
name [name] VARCHAR(255)
passwdHash [passwdHash] CHAR(32)
loginEnabled [loginEnabled] BIT
passwordExpired [passwordExpired] BIT
has assignment / is assigned to
modified / was modified by
owns / is businessOwner of
has membership / is membership of
has primaryGroup / is primaryGroup of
has businessOwner / is businessOwner of
Groups [Groups]
PK gid [gid] INTEGER
name [name] VARCHAR(255)
has / is of
Privileges [Privileges]
PK id [id] INTEGER
name [name] CHAR(20)
has / is of
PrivilegeLists [PrivilegeLists]
PK,FK1 pid [pid] INTEGER
PK,FK2 gid [gid] INTEGER
enabled [enabled] BIT

3 Architectural Design

3.1 Program Structure

The program structure chosen for this application is a data−centered structure. The database resides at the center of this architecture. The components that make up this application will frequently read from this database, and write to this database. The diagram below illustrates this concept.

3.1.1 Architectural Diagram



4 Component–Level Design

This section describes the design of each component in detail. For each component, a short narrative will be given. This will be followed by a description of a possible user interface. A basic algorithm will be explained, followed by an activity diagram (if appropriate).

4.1 Description of Add Group

This component will allow a user to add a group to the system. Groups are made up of users who share the same privileges to the system.

4.1.1 Processing Narrative

A user wishes to add a group to the list of groups. In order to do this, the user has to have been authenticated and authorized. The user must go to the *User Management* tab on the main screen, and click the button *Add Group*. The user then fills in required and optional information (described in the interface description below).

4.1.2 Interface Description

The Add Group function is activated when the user clicks on the *Add Group* button (located in the *User Management* tab on the main screen. When this button is selected, the following interface will be visible on the screen. There will be a text box to enter the name of the group, and its description. There will be a control (possibly a drop down menu) to select a module. The modules are: Watches, Inventory, Requirements, Configuration, and User Management. It may be possible to configure this list. There will be a control next to this, which contains checkboxes for privileges. The privileges are: View, Create, Modify, and Delete. There will be a *Save* command button next to this. Essentially, the user will select a module in the drop down box, check the desired privileges for that module, and select the *Save* button. This will be repeated for each module.

There will also be an option to add users to this group. There will be a control (possibly a drop down menu) populated with all of the users in the database. To add a user to a group, one would select a user from the control, and click on the *Add* command button located nearby.

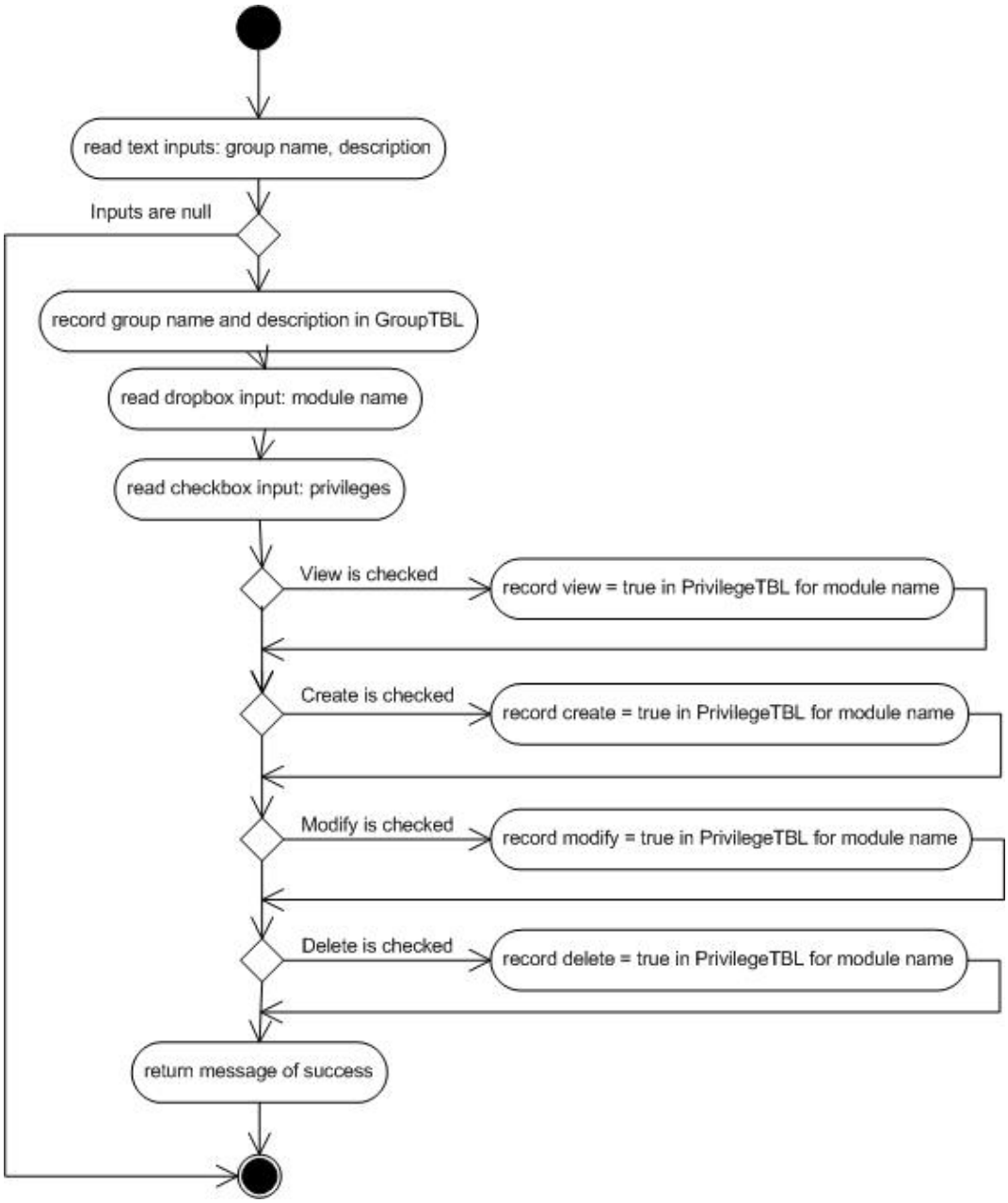
4.1.3 Component Processing Detail

4.1.3.1 Algorithmic Model

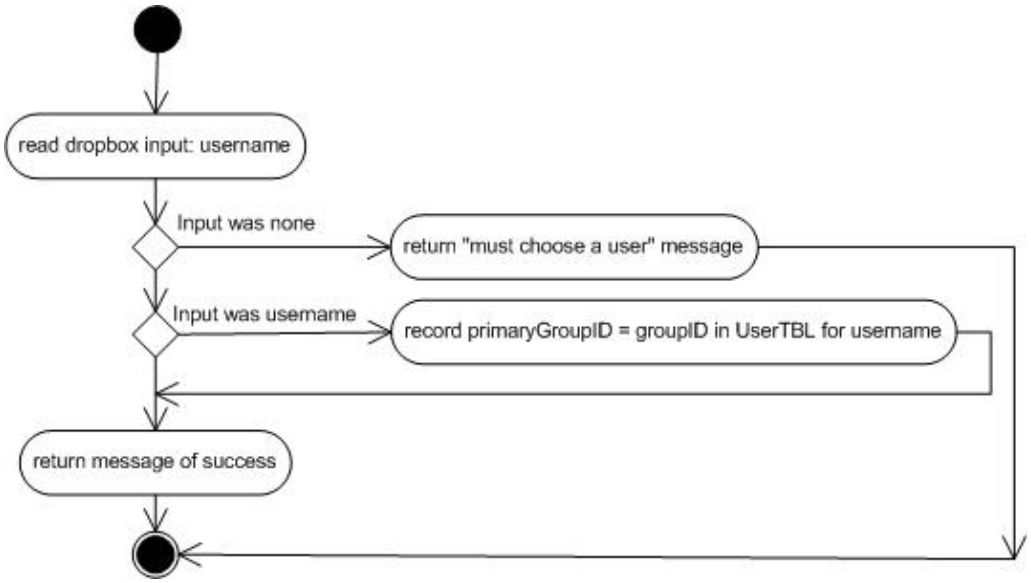
```
        In: GroupName
        In: Description
        In: ModuleName = Watches, Inventory, Requirements, Configuration, User Management
        In: View = true, false
        In: Create = true, false
        In: Modify = true, false
        In: Delete = true, false
        In: Users = None, User1 to UserN
        Out: message of success or message of failure

Save( In: GroupName, Description, ModuleName, Create, Modify, Delete, View
      Out: message )
        validate inputs
        On Error: send message of failure and exit the routine
```

```
        record to GroupTBL -> GroupName
        record to GroupTBL -> Description
        X = ModuleName.Value
        if View is true:
                record to PrivilegeTBL (where Module is X) -> View = true
        if Create is true:
                record to PrivilegeTBL (where Module is X) -> Create = true
        if Modify is true:
                record to PrivilegeTBL (where Module is X) -> Modify = true
        if Delete is true:
                record to PrivilegeTBL (where Module is X) -> Delete = true
        if no errors occurred:
                send message of success

Add( In: Users
     Out: message )
        validate inputs
        On Error: send message of failure and exit the routine
        if Users.Value is None:
                send message and exit the routine
        else
                lookup from GroupTBL -> GroupID
                TheID = GroupID
                record to UserTBL -> PrimaryGroupID = TheID
        if no errors occurred:
                send message of success
```

4.1.3.2 Activity diagram – Save( )


read text inputs: group name, description
Inputs are null
record group name and description in GroupTBL
read dropbox input: module name
read checkbox input: privileges
View is checked
record view = true in PrivilegeTBL for module name
Create is checked
record create = true in PrivilegeTBL for module name
Modify is checked
record modify = true in PrivilegeTBL for module name
Delete is checked
record delete = true in PrivilegeTBL for module name
return message of success

4.1.3.3 Activity diagram – Add( )


read dropbox input: username
Input was none
return "must choose a user" message
Input was username
record primaryGroupID = groupID in UserTBL for username
return message of success

4.2 Description of Add User

This component will allow the current user to add a user object to the system. The newly created user object will be assigned to a group. All users must be assigned to a group for security.

4.2.1 Processing Narrative

A user wishes to add a user object to the list of user objects. In order to do this, the user has to have been authenticated and authorized. The user must go to the *User Management* tab on the main screen, and click the button *Add User*. The user then fills in required and optional information (described in the interface description below).

4.2.2 Interface Description

The Add User function is activated when the current user clicks on the *Add User* button (located in the *User Management* tab on the main screen. When this button is selected, the following interface will be visible on the screen. There will be text boxes for the Login Name and User Name. The Login Name is the name that the user will use to log in to the system. The User Name is the name that the program will use to refer to the user. For example, a Login Name could be "jdoe" while a User Name might be "Jane Doe". There will also be a text box to specify the password that the new user will start off with. This password can be changed by the user at any time during login.

There will be a control (possibly a drop down box) populated with a list of groups. Once a group is selected for the new user object, a *Save* command button can be selected.

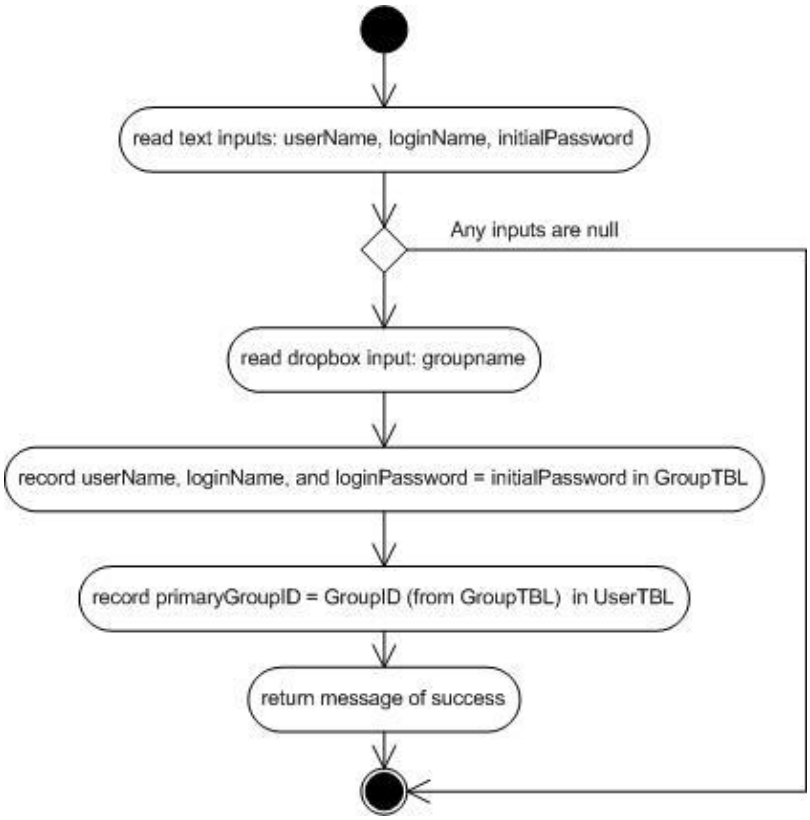
4.2.3 Component Processing Detail

4.2.3.1 Algorithmic Model

```
In: LoginName
In: UserName
In: InitialPassword
In: Group
```

```
        Out: message of success or message of failure

Save( In: LoginName, UserName, InitialPassword, Group
      Out: message )
        validate inputs
        On Error: send message of failure and exit the routine
        record to UserTBL -> LoginName
        record to UserTBL -> UserName
        record to UserTBL -> LoginPassword = InitialPassword
        lookup from GroupTBL (where name is Group.Value) -> GroupID
        TheID = GroupID
        record to UserTBL -> PrimaryGroupID = TheID
        if no errors occurred:
                send message of success
```

4.2.3.2 Activity diagram – Save( )


read text inputs: userName, loginName, initialPassword
Any inputs are null
read dropbox input: groupname
record userName, loginName, and loginPassword = initialPassword in GroupTBL
record primaryGroupID = GroupID (from GroupTBL) in UserTBL
return message of success

4.3 Description of View Group List

This component will allow a user to view a list of all the groups currently stored in the system.

4.3.1 Processing Narrative

A user wishes to view the list of groups. In order to do this, the user has to have been authenticated and authorized. The user must go to the *User Management* tab on the main screen, and select the *View Group List* button. The user then fills in required and optional information (described in the interface description below).

4.3.2 Interface Description

The View Group List function is activated when the user clicks on the *View Group List* button (located in the *User Management* tab on the main screen. When this button is selected, the following interface will be visible on the screen. A list will be displayed containing the Group Name, and the Description. There will be a way to select an entry in this list. There will be three command buttons, one to view the details of the selected entry, one to modify it, and one to delete it. If the user does not have the correct privilege for one of the buttons, that button will be disabled.

4.3.3 Component Processing Detail

4.3.3.1 Algorithmic Model

```
        In: possible selection from list of groups
        Out: a list of the groups that are currently in the system

ViewGroupList( )
        On Error: send message of failure and exit the routine
        Populate a list with the following information:
                Name and Description for each GroupID in GroupTBL

ViewGroupDetails( In: selection
                  Out: message )
        This algorithm is described in the View Group component.

Modify( In: selection
        Out: message )
        This algorithm is described in the Modify Group component.

Delete( In: selection
        Out: message )
        This algorithm is described in the Delete  Group component.
```

4.4 Description of View User List

This component will allow the current user to view a list of all the user objects currently stored in the system.

4.4.1 Processing Narrative

The current user wishes to view the list of groups. In order to do this, the user has to have been authenticated and authorized. The user must go to the *User Management* tab on the main screen, and select the *View User List* button. The user then fills in required and optional information (described in the interface description below).

4.4.2 Interface Description

The View User List function is activated when the user clicks on the *View User List* button (located in the *User Management* tab on the main screen. When this button is selected, the following interface will be visible on the screen. A list will be displayed containing (for each user on the list) the User Name, the Login Name and the name of the group. There will be a way to select an entry in this list. There will be three command buttons, one to view the details of the selected entry, one to modify it, and one to delete it. If the current user does not have the correct privilege for one of the buttons, that button will be disabled.

4.4.3 Component Processing Detail

4.4.3.1 Algorithmic Model

```
        In: possible selection from list of user objects
        Out: a list of the user objects that are currently in the system

ViewUserList( )
        On Error: send message of failure and exit the routine
        Populate a list with the following information:
                UserName and LoginName for each UserID in UserTBL
                GroupName (where Group.ID is User.ID) from  GroupTBL

ViewUserDetails( In: selection
                 Out: message )
        This algorithm is described in the View User component description.

Modify( In: selection
        Out: message )
        This algorithm is described in the Modify User component description.

Delete( In: selection
        Out: message )
        This algorithm is described in the Delete User component description.
```

4.5 Description of View Group

This component will allow a user to view the details of a group that is currently in the system.

4.5.1 Processing Narrative

A user wishes to view the details of a group. In order to do this, the user has to have been authenticated and authorized. There is more than one way the user can do this:

- The user can go to the *User Management* tab on the main screen, and click the button *View Group List*. The user may then select a group from the list, and click *View Details*.
- If the user is in the *View User* section, they can select the *View Group Details* button that is next to the group name description. (This is described in the *View User* component description.

4.5.2 Interface Description

The name of the group and its description will be displayed. There will also be a list of modules (populated from the database into a control structure). All of the privileges for each module will also be displayed. One possible way this could be achieved is: a list box will store the names of the modules. There will be an un−editable checkbox control next to it containing the privileges. When a module is selected, its privileges become *checked* in the list of privileges.

There will also be a list of users that are currently in the group. This will also be put into a list structure that has been populated from the database. There will be a command button called *ViewUserDetails* that will take you to the View User component for the selected user. (See the View User component description for more details.)

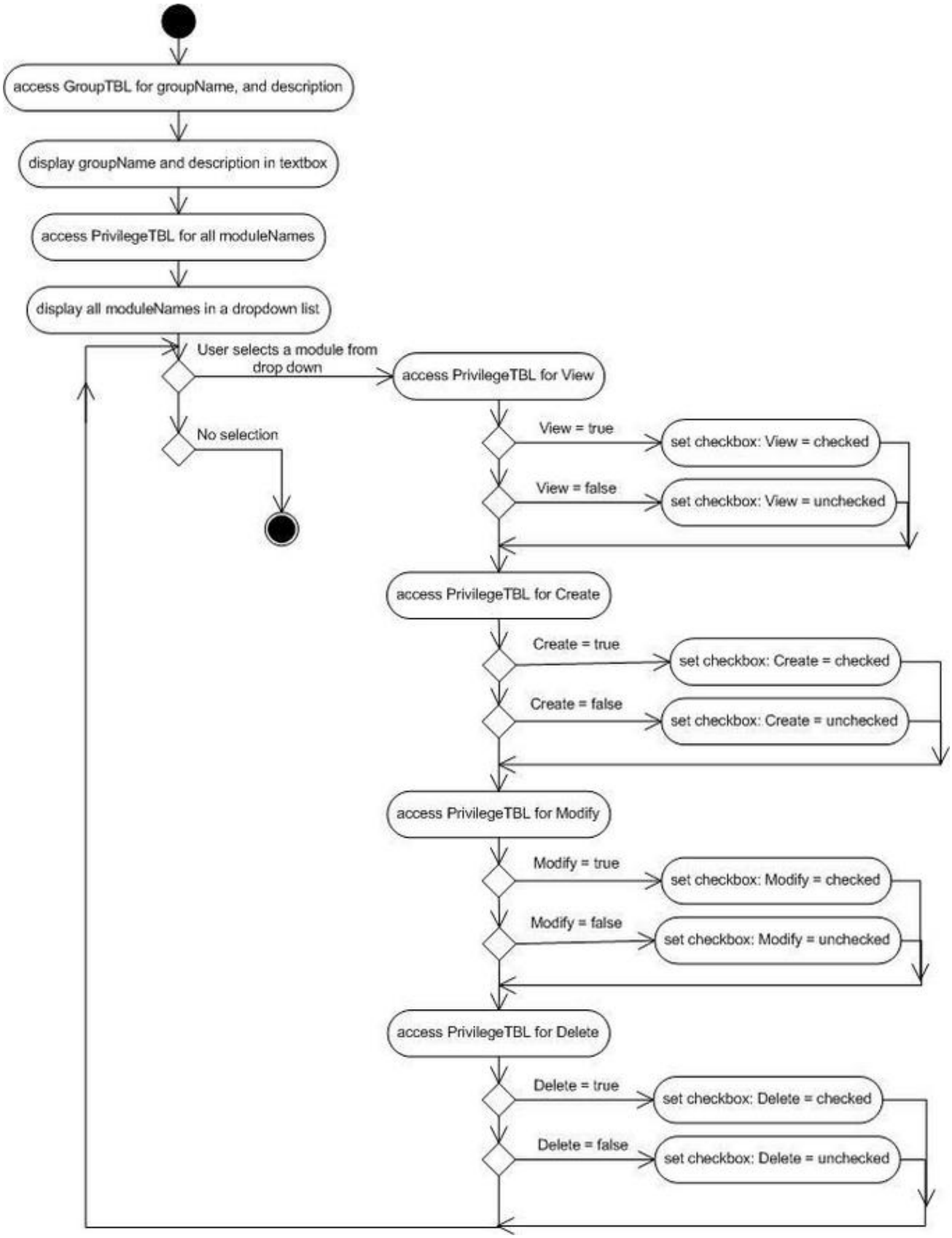
There will be a command button called *Modify*. It will be enabled if the user has the modify privilege for user management. This will allow the user to modify the attributes of this group. (See the Modify Group component description for more details.) There will also be a command button called *Delete*. It will be enabled if the user has the delete privilege for user management. (See the Delete Group component description for more details.)

4.5.3 Component Processing Detail

4.5.3.1 Algorithmic Model

```
        In: the GroupID from the GroupTBL for a selected group
        In: possible selection from list of modules
        In: possible selection from list of users

ViewGroup( In: GroupID )
        On Error: send message of failure and exit the routine
        lookup from GroupTBL  -> GroupName
        lookup from GroupTBL -> Description
        Populate a list with the following information:
                the UserName for each ID in the UserTBL
        Populate a list with the following information:
                the ModuleName for each module in the PrivilegeTBL
        Populate a check box list with the following information:
                a Create box (set to unchecked)
                a Modify box (set to unchecked)
                a View box (set to unchecked)
                a Delete box (set to unchecked)
        If a ModuleName is selected:
                lookup from PrivilegeTBL (where Module is ModuleName -> Create.Value
                if Create.Value is true: the Create box is checked
                lookup from PrivilegeTBL (where Module is ModuleName -> Modify.Value
                if Modify.Value is true: the Modify box is checked
                lookup from PrivilegeTBL (where Module is ModuleName -> View.Value
                if View.Value is true: the View box is checked
                lookup from PrivilegeTBL (where Module is ModuleName -> Delete.Value
                if Delete.Value is true: the Delete box is checked

ViewUserDetails( In: UserID from selection )
        This will direct the user to the ViewUser module

Modify( GroupID )
        This algorithm is described in the Modify Group component description.

Delete( GroupID )
        This algorithm is described in the Delete Group component description.
```

4.5.3.2 Activity diagram – ViewGroup( )


access GroupTBL for groupName, and description
display groupName and description in textbox
access PrivilegeTBL for all moduleNames
display all moduleNames in a dropdown list
User selects a module from drop down
No selection
access PrivilegeTBL for View
View = true
set checkbox: View = checked
View = false
set checkbox: View = unchecked
access PrivilegeTBL for Create
Create = true
set checkbox: Create = checked
Create = false
set checkbox: Create = unchecked
access PrivilegeTBL for Modify
Modify = true
set checkbox: Modify = checked
Modify = false
set checkbox: Modify = unchecked
access PrivilegeTBL for Delete
Delete = true
set checkbox: Delete = checked
Delete = false
set checkbox: Delete = unchecked

4.6 Description of View User

This component will allow the current user to view the details of a user object that is currently in the system.

4.6.1 Processing Narrative

A user wishes to view the details of a user object. In order to do this, the user has to have been authenticated and authorized. There is more than one way the user can do this:

- The user can go to the *User Management* tab on the main screen, and click the button *View User List*. The user may then select a user object from the list, and click *View Details*.
- If the user is in the *View Group* section, they can select the *View User Details* button after selecting an entry in the list of user objects. (This is described in the *View Group* component description.

4.6.2 Interface Description

The user's name, the user's login name, and the name of the group the user is currently in will be displayed. Next to the name of the group there will be a command button called *ViewGroupDetails* that will take you to the View Group component for this user. (See the View Group component description for more details).

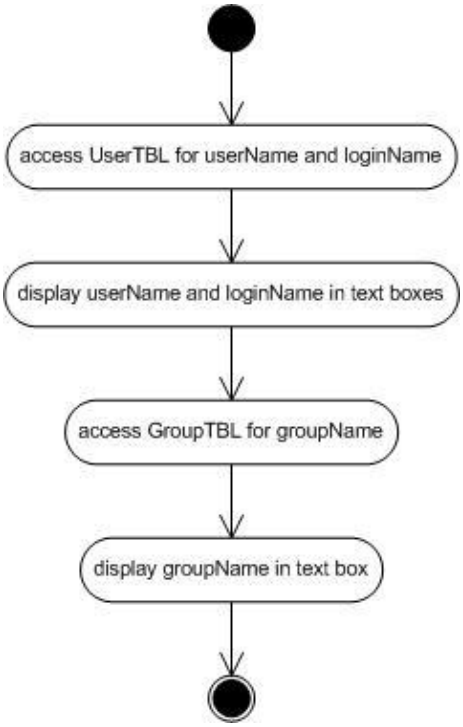
There will be a command button called *Modify*. It will be enabled if you have the modify privilege for user management. This will allow you to modify the attributes of this user object. (See the Modify Object component description for more details.) There will also be a command button called *Delete*. It will be enabled if you have the delete privilege for user management. (See the Delete User component description for more details.)

4.6.3 Component Processing Detail

4.6.3.1 Algorithmic Model

```
        In: the UserID from the UserTBL for a selected user
        In: possible selection from list of modules
        In: possible selection from list of users

ViewUser( In: UserID )
        On Error: send message of failure and exit the routine
        lookup from UserTBL  -> UserName
        lookup from UserTBL -> LoginName
        lookup from UserTBL -> GroupID
        lookup from GroupTBL -> GroupName

ViewGroupDetails( In: GroupID )
        This will direct the user to the ViewUser module

Modify( UserID )
        This algorithm is described in the Modify User component description.

Delete( UserID )
        This algorithm is described in the Delete User component description.
```

4.6.3.2 Activity diagram – ViewUser( )


access UserTBL for userName and loginName
display userName and loginName in text boxes
access GroupTBL for groupName
display groupName in text box

4.7 Description of Modify Group

This component will allow a user to modify the details of a group that is currently in the system.

4.7.1 Processing Narrative

A user wishes to modify the details of a group. In order to do this, the user has to have been authenticated and authorized. There is more than one way to get to the modification interface:

- The user can go to the *User Management* tab on the main screen, and click the button *View Group List*. The user may select the *Modify* command button.
- The user can go to the *User Management* tab on the main screen, and click the button *View Group List*. The user may then select a group from the list, and click *View Details*. (A user can also get to *View Details* through the *View User* section (described in the *View User* component description.) From the *View Details* screen, the user may select the *Modify* command button.

4.7.2 Interface Description

The name of the group and its description will be displayed. These are editable. There will also be a list of modules (populated from the database into a control structure). All of the privileges for each module will also be displayed. One possible way this could be achieved is: a list box will store the names of the modules. There will be an un−editable checkbox control next to it containing the privileges. When a module is selected, its privileges become *checked* in the list of privileges. The user will be able to edit these checked boxes and save the changes using the *Save* button.

There will also be a list of user objects that are currently in the group. This will also be put into a list structure that has been populated from the database. The current user will be able to select one of these user objects from the list. A selected user object can be deleted from the group when the current user selects the *Remove* command button. When a user is removed from a group, they are put in the default group.

There will also be an option to add users to this group. There will be a control (possibly a drop down menu) populated with all of the users in the database who aren't already in the group. To add a user to a group, one would select a user from the control, and click on the *Add* command button located nearby.

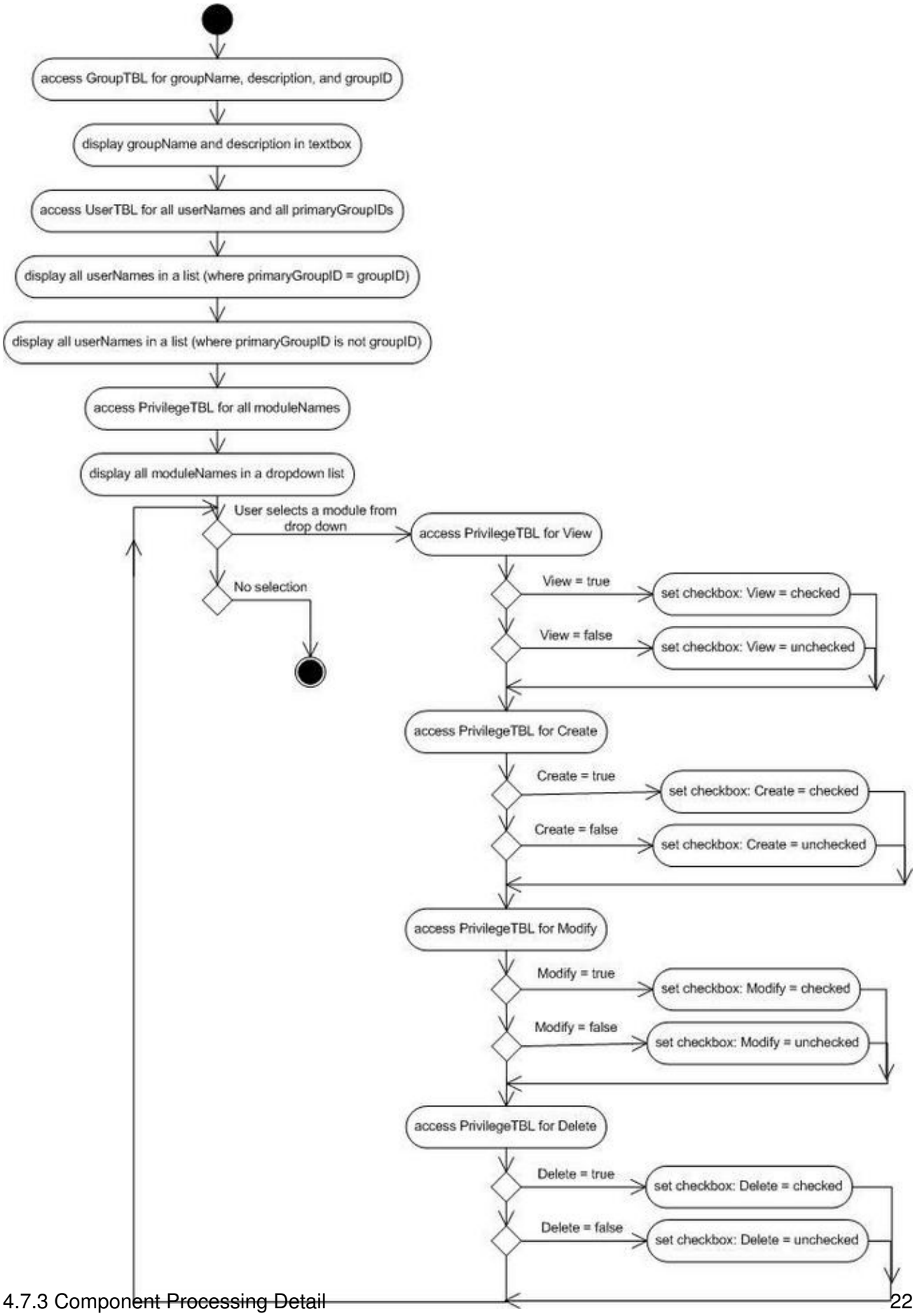
4.7.3 Component Processing Detail

4.7.3.1 Algorithmic Model

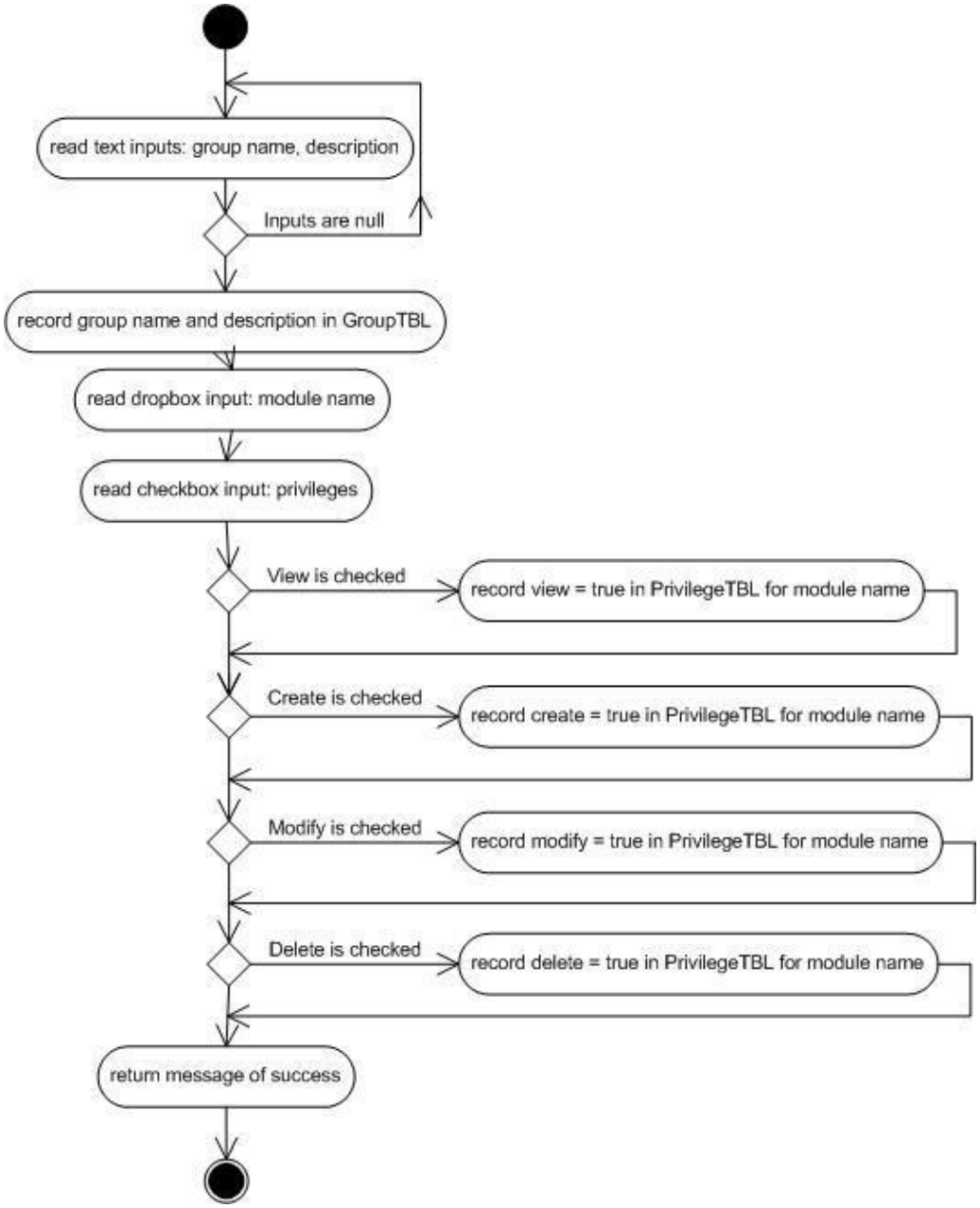
```
        In: the GroupID from the GroupTBL for a selected group
        In: SelA = possible selection from list of users in group
        In: SelB = possible selection from list of user not in group
        In: GroupName
        In: GroupDescription
        In: Create = true, false
        In: View = true, false
        In: Modify = true, false
        In: Delete = true, false

ModifyGroup( In: GroupID )
        On Error: send message of failure and exit the routine
        lookup from GroupTBL  -> GroupName
        lookup from GroupTBL -> Description
        Populate a list with the following information:
                the UserName for each ID in the UserTBL (For users in group)
        Populate a list with the following information:
                the UserName for each ID in the UserTBL (For users not in group)
        Populate a list with the following information:
                the ModuleName for each module in the PrivilegeTBL
        Populate a check box list with the following information:
                a Create box (set to unchecked)
                a Modify box (set to unchecked)
                a View box (set to unchecked)
                a Delete box (set to unchecked)
        If a ModuleName is selected:
                lookup from PrivilegeTBL (where Module is ModuleName -> Create.Value
                if Create.Value is true: the Create box is checked
                lookup from PrivilegeTBL (where Module is ModuleName -> Modify.Value
                if Modify.Value is true: the Modify box is checked
                lookup from PrivilegeTBL (where Module is ModuleName -> View.Value
                if View.Value is true: the View box is checked
                lookup from PrivilegeTBL (where Module is ModuleName -> Delete.Value
                if Delete.Value is true: the Delete box is checked

Save( In: GroupName, Description, ModuleName, Create, Modify, Delete, View
      Out: message )
        validate inputs
        On Error: send message of failure and exit the routine
        record to GroupTBL -> GroupName
        record to GroupTBL -> Description
        X = ModuleName.Value
        if View is true:
                record to PrivilegeTBL (where Module is X) -> View = true
        if Create is true:
                record to PrivilegeTBL (where Module is X) -> Create = true
```

```
        if Modify is true:
                record to PrivilegeTBL (where Module is X) -> Modify = true
        if Delete is true:
                record to PrivilegeTBL (where Module is X) -> Delete = true
        if no errors occurred:
                send message of success

Add( In: SelB
     Out: message )
        validate inputs
        On Error: send message of failure and exit the routine
        if SelB.Value is None:
                send message and exit the routine
        else
                lookup from GroupTBL -> GroupID
                TheID = GroupID
                record to UserTBL -> PrimaryGroupID = TheID
        if no errors occurred:
                send message of success

Remove( In: SelA
         Out: message )
        validate inputs
        On Error: send message of failure and exit the routine
        if SelA.Value is (global) UserName (yourself)
                send message and exit the routine
        else
                record to UserTBL -> PrimaryGroupID = DefaultID
                refresh list of users that SelA came from
        if no errors occurred:
                send message of success
```

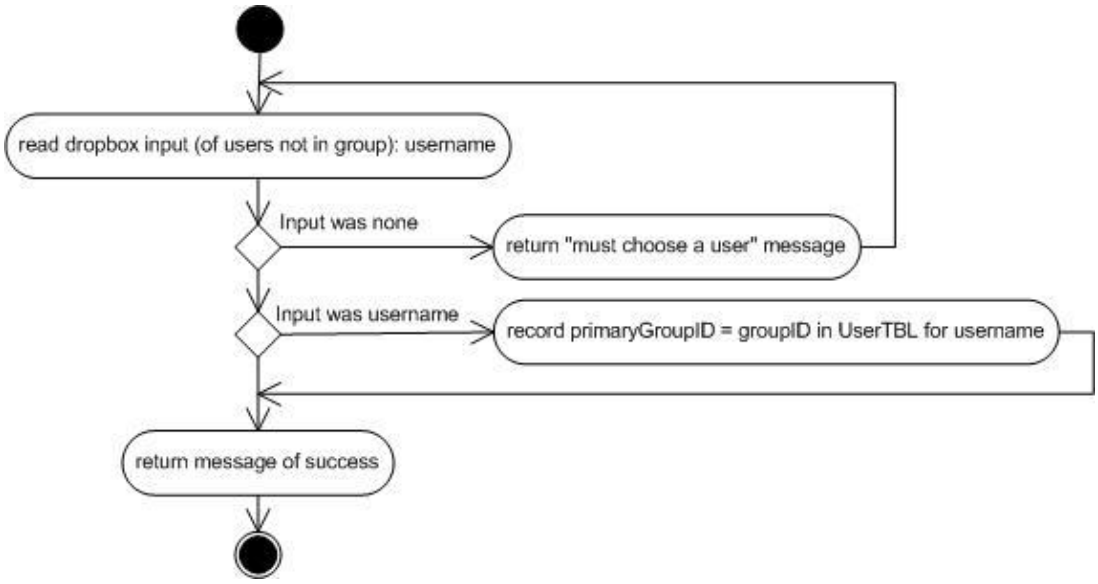
4.7.3.2 Activity diagram – ModifyGroup( )


access GroupTBL for groupName, description, and groupID
display groupName and description in textbox
access UserTBL for all userNames and all primaryGroupIDs
display all userNames in a list (where primaryGroupID = groupID)
display all userNames in a list (where primaryGroupID is not groupID)
access PrivilegeTBL for all moduleNames
display all moduleNames in a dropdown list
User selects a module from drop down
No selection
access PrivilegeTBL for View
View = true
set checkbox: View = checked
View = false
set checkbox: View = unchecked
access PrivilegeTBL for Create
Create = true
set checkbox: Create = checked
Create = false
set checkbox: Create = unchecked
access PrivilegeTBL for Modify
Modify = true
set checkbox: Modify = checked
Modify = false
set checkbox: Modify = unchecked
access PrivilegeTBL for Delete
Delete = true
set checkbox: Delete = checked
Delete = false
set checkbox: Delete = unchecked

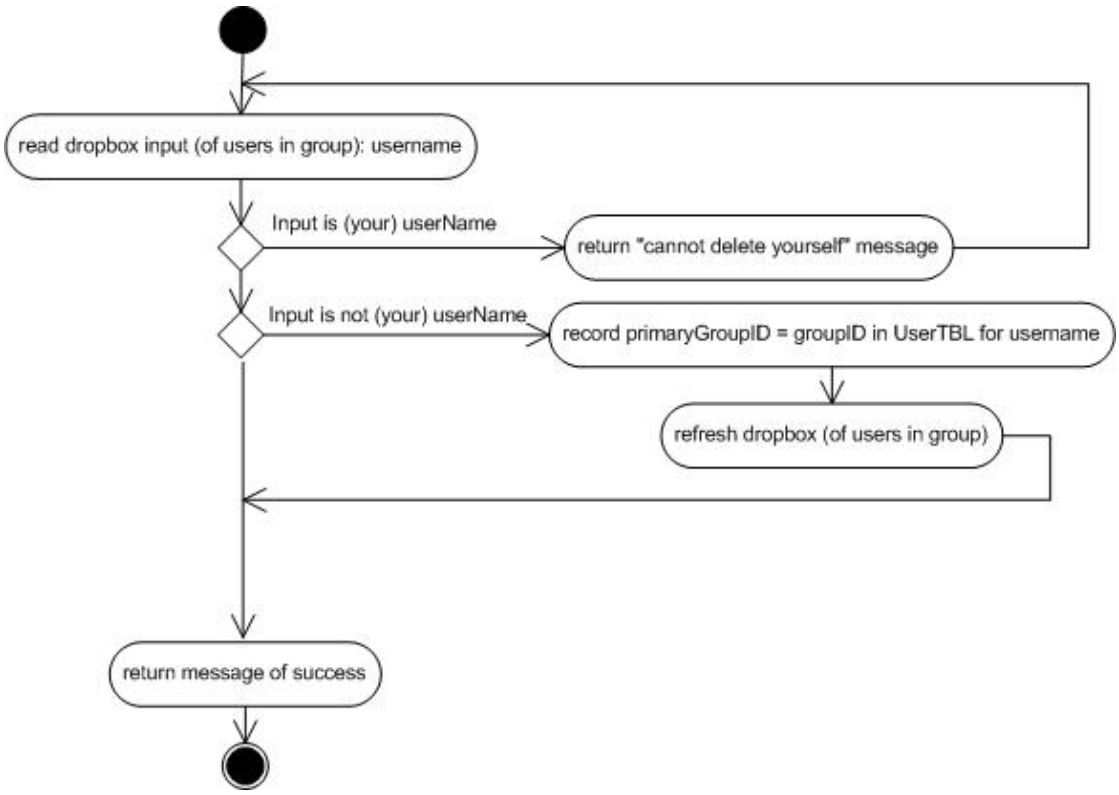
4.7.3.3 Activity diagram – Save( )


read text inputs: group name, description
Inputs are null
record group name and description in GroupTBL
read dropbox input: module name
read checkbox input: privileges
View is checked
record view = true in PrivilegeTBL for module name
Create is checked
record create = true in PrivilegeTBL for module name
Modify is checked
record modify = true in PrivilegeTBL for module name
Delete is checked
record delete = true in PrivilegeTBL for module name
return message of success

4.7.3.4 Activity diagram – Add( )


read dropbox input (of users not in group): username
Input was none
return "must choose a user" message
Input was username
record primaryGroupID = groupID in UserTBL for username
return message of success

4.7.3.5 Activity diagram – Remove( )


read dropbox input (of users in group): username
Input is (your) userName
return "cannot delete yourself" message
Input is not (your) userName
record primaryGroupID = groupID in UserTBL for username
refresh dropbox (of users in group)
return message of success

4.8 Description of Modify User

This component will allow the current user to modify the details of a user object that is currently in the system.

4.8.1 Processing Narrative

The current user wishes to modify the details of a user object. In order to do this, the current user has to have been authenticated and authorized. There is more than one way to get to the modification interface:

- The user can go to the *User Management* tab on the main screen, and click the button *View User List*. The user may select the *Modify* command button.
- The user can go to the *User Management* tab on the main screen, and click the button *View User List*. The user may then select a user object from the list, and click *View Details*. (A user can also get to *View Details* through the *View Group* section (described in the *View Group* component description.) From the *View Details* screen, the user may select the *Modify* command button.

4.8.2 Interface Description

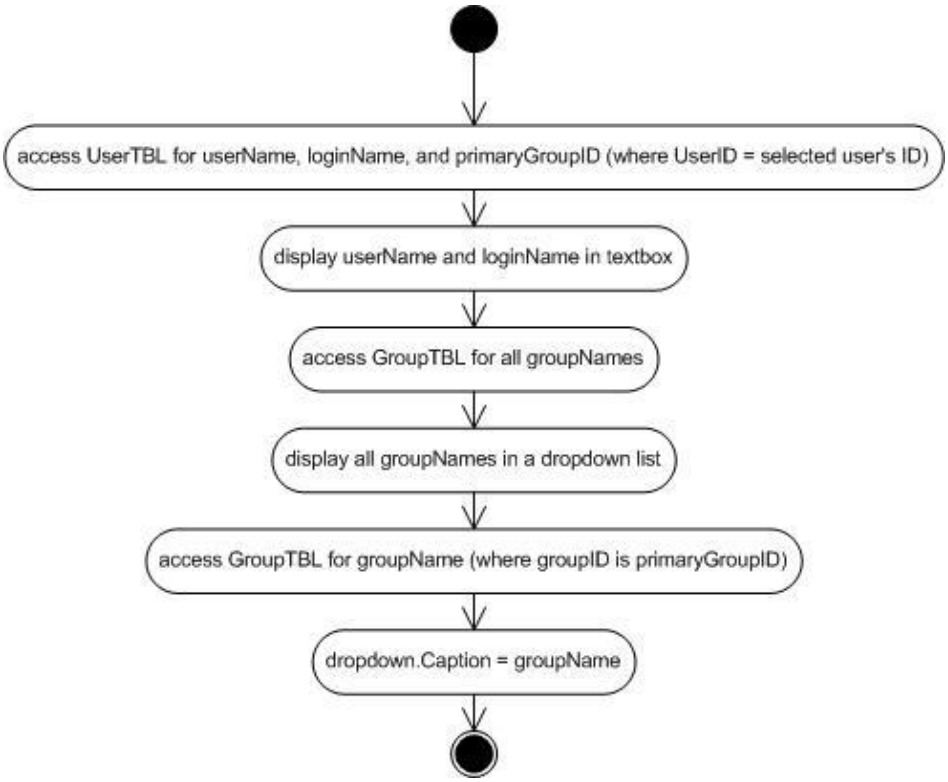
The name of the user and the user's log in name will be displayed. These will be editable. There will be a control (possibly a drop down box) populated with a list of groups. The group that the user belongs to will be shown as the caption. You can change the group that the user is in, by selecting a group from the box. All changes will be saved by selecting the *Save* command button.

4.8.3 Component Processing Detail

4.8.3.1 Algorithmic Model

```
        In: the UserID from the UserTBL for a selected user
        In: possible selection from list of groups
        In: UserName
        In: LoginName
        In: Group

ModifyUser( In: UserID )
        On Error: send message of failure and exit the routine
        lookup from UserTBL  -> UserName
        lookup from UserTBL -> LoginName
        lookup from UserTBL -> PrimaryGroupID
        Populate a drop box  with the following information:
                the GroupName for each ID in the GroupTBL
        Lookup from GroupTBL -> GroupName where GroupID is PrimaryGroupID
        Make caption of drop box = GroupName

Save( In: UserName, LoginName, Group
      Out: message )
        validate inputs
        On Error: send message of failure and exit the routine
        record to UserTBL -> LoginName
        record to UserTBL -> UserName
        lookup from GroupTBL (where name is Group.Value) -> GroupID
        TheID = GroupID
        record to UserTBL -> PrimaryGroupID = TheID
        if no errors occurred:
                send message of success
```

4.8.3.2 Activity diagram – Save( )


access UserTBL for userName, loginName, and primaryGroupID (where UserID = selected user's ID)
display userName and loginName in textbox
access GroupTBL for all groupNames
display all groupNames in a dropdown list
access GroupTBL for groupName (where groupID is primaryGroupID)
dropdown.Caption = groupName

4.9 Description of Delete Group

This component will allow a user to delete a group from the system.

4.9.1 Processing Narrative

A user wishes to delete a group from the program. In order to do this, the user has to have been authenticated and authorized. There is more than one way to access the delete interface:

- The user can go to the *User Management* tab on the main screen, and select the button *View Group List*. The user may then select a group from the list, and select *Delete*.
- The user can go to the *User Management* tab on the main screen, and select the button *View Group List*. The user can select a group from the list, and select *View Details*. From this screen, there will also be a *Delete* command button.

4.9.2 Interface Description

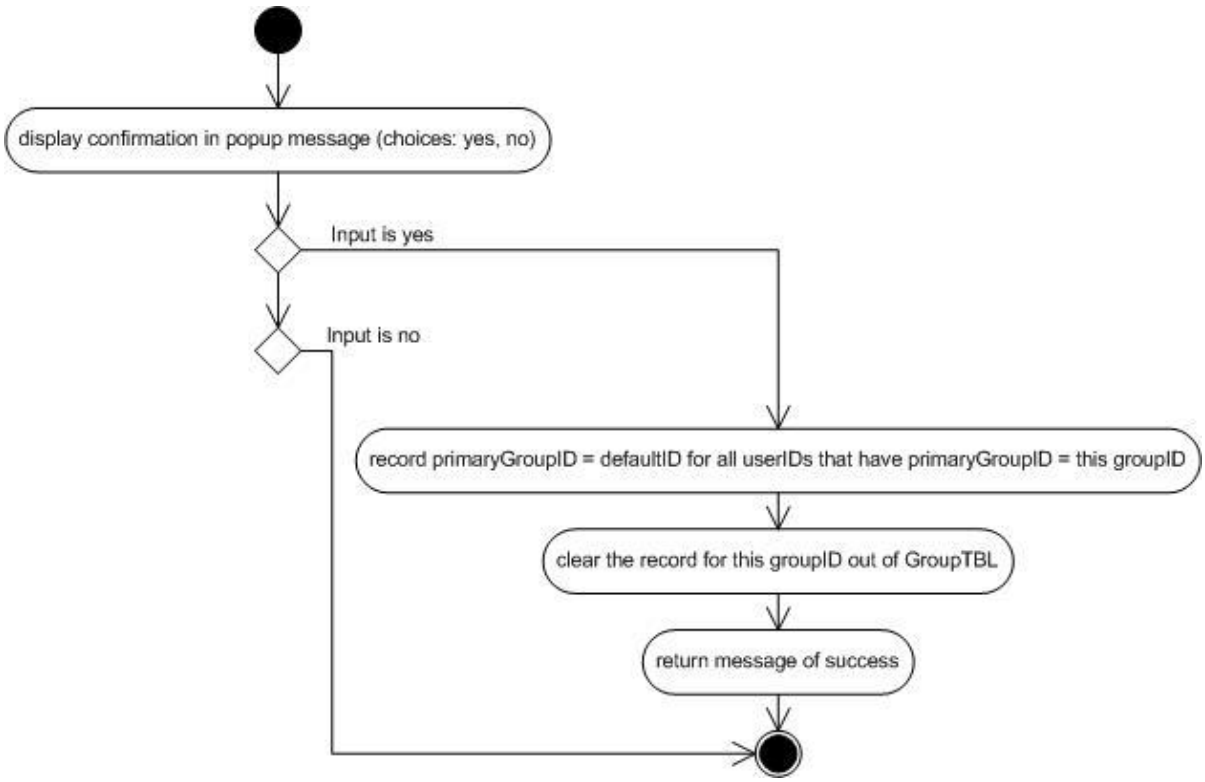
When a *Delete* button is selected for a group, the following interface will be visible on the screen. A confirmation message box will appear on the screen. If the user selects *Yes*, the group will be deleted. If the user selects *No*, nothing will happen.

4.9.3 Component Processing Detail

4.9.3.1 Algorithmic Model

```
        In: Delete = yes, no
        Out: message of success or message of failure

Yes( Out: message )
        On Error: send message of failure and exit the routine
        for each user in UserTBL that has PrimaryGroupId = GroupId:
                record to UserTBL -> PrimaryGroupId = default
        clear the record out of the GroupTBL
        if no errors occurred:
                send message of success

No( )
        do nothing
```

4.9.3.2 Activity diagram – DeleteGroup( )


display confirmation in popup message (choices: yes, no)
Input is yes
Input is no
record primaryGroupID = defaultID for all userIDs that have primaryGroupID = this groupID
clear the record for this groupID out of GroupTBL
return message of success

4.10 Description of Delete User

This component will allow the current user to delete a user object from the system.

4.10.1 Processing Narrative

The current user wishes to delete a user object from the program. In order to do this, the user has to have been authenticated and authorized. There is more than one way to access the delete interface:

- The user can go to the *User Management* tab on the main screen, and select the button *View User List*. The user may then select a user object from the list, and select *Delete*.

- The user can go to the *User Management* tab on the main screen, and select the button *View User List*. The user can select a user object from the list, and select *View Details*. From this screen, there will also be a *Delete* command button.

4.10.2 Interface Description

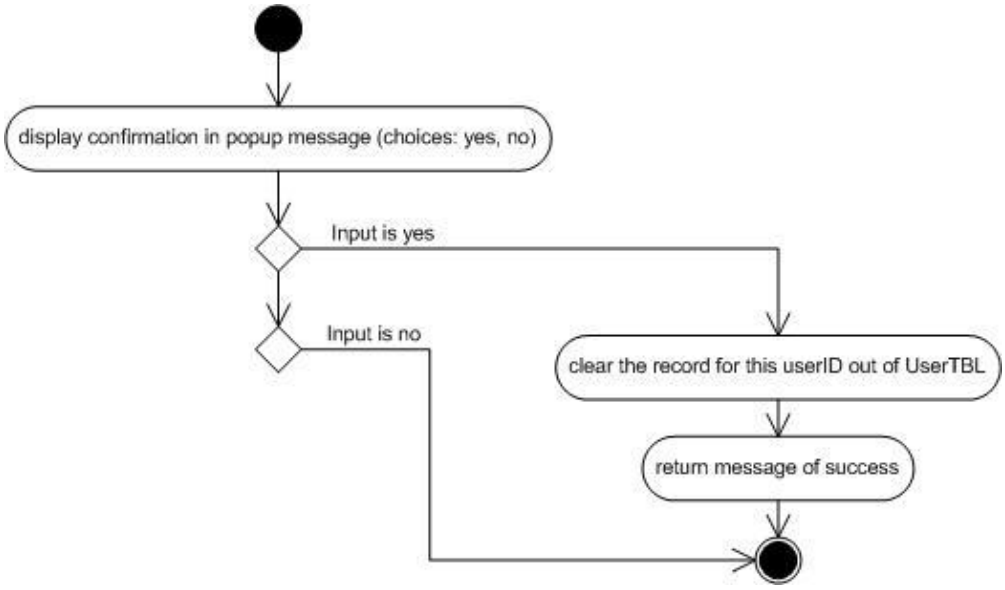
When a *Delete* button is selected for a user object, the following interface will be visible on the screen. A confirmation message box will appear on the screen. If the user selects *Yes*, the user object will be deleted. If the user selects *No*, nothing will happen. Users will not be able to delete themselves.

4.10.3 Component Processing Detail

4.10.3.1 Algorithmic Model

```
        In: Delete = yes, no
        Out: message of success or message of failure

Yes( Out: message )
        On Error: send message of failure and exit the routine
        clear the record out of the UserTBL
        if no errors occurred:
                send message of success

No( )
        do nothing
```

4.10.3.2 Activity diagram – DeleteUser( )


display confirmation in popup message (choices: yes, no)
Input is yes
Input is no
clear the record for this userID out of UserTBL
return message of success

4.11 Description of Log In

This component will allow the user to log in to the system and become authenticated.

4.11.1 Processing Narrative

A user wishes to log in to the system. First, the user must launch the application. The user must then enter their name and password. If this information is correct, the user will be able to use the tracking tool.

4.11.2 Interface Description

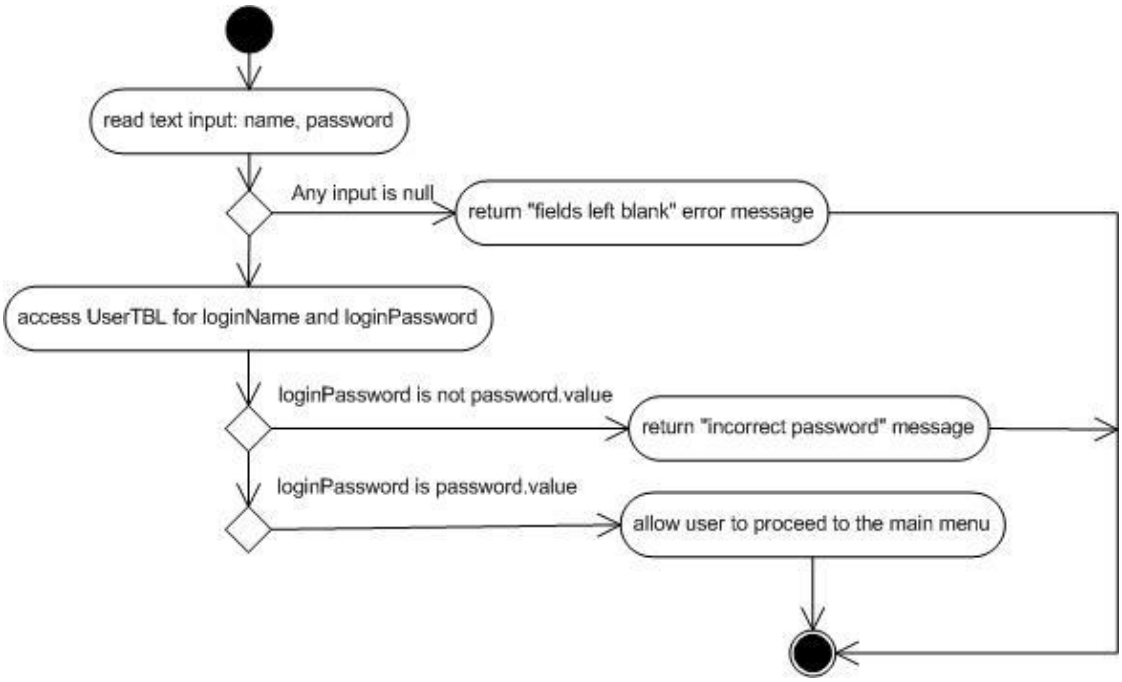
There will be text fields for the user to input their username and password. The password field will be masked for security. There will be a command button to submit this information. If the information is correct, the user will see a message of success. If the information is incorrect, the user will receive a message of failure. There should also be a command button to let the user change their password. (See Change Password component description.)

4.11.3 Component Processing Detail

4.11.3.1 Algorithmic Model

```
        In: Name
        In: Password
        Out: message of success or message of failure

Submit( In: Name, Password
        Out: message )
        validate inputs
        On Error: send message of failure and exit the routine
        lookup from UserTBL (where Name.Value = LoginName) -> LoginName
        lookup from UserTBL (where Password.Value = LoginPassword) -> LoginPassword
        CorrectPassword = LoginPassword
        if CorrectPassword is not Password.Value
                send message of failure and exit the routine
        if no errors occurred:
                send message of success
```

4.11.3.2 Activity diagram – LogIn( )


read text input: name, password
Any input is null
return "fields left blank" error message
access UserTBL for loginName and loginPassword
loginPassword is not password.value
return "incorrect password" message
loginPassword is password.value
allow user to proceed to the main menu

4.12 Description of Change Password

This component will allow the user to change their password during the log in stage.

4.12.1 Processing Narrative

A user wishes to change their password. First, the user must launch the application. The first screen will be a login interface. On this interface there is a command button called *Change Password*. The user selects this button, which brings up the change password interface described below.

4.12.2 Interface Description

There are text boxes for the user's login name, current password, new password, and confirmation of new password. When the user is finished, they will select the *Submit* button. If the information is correct, they will receive a confirmation message. If the information is incorrect, they will receive a message of failure, and their password will not be altered.

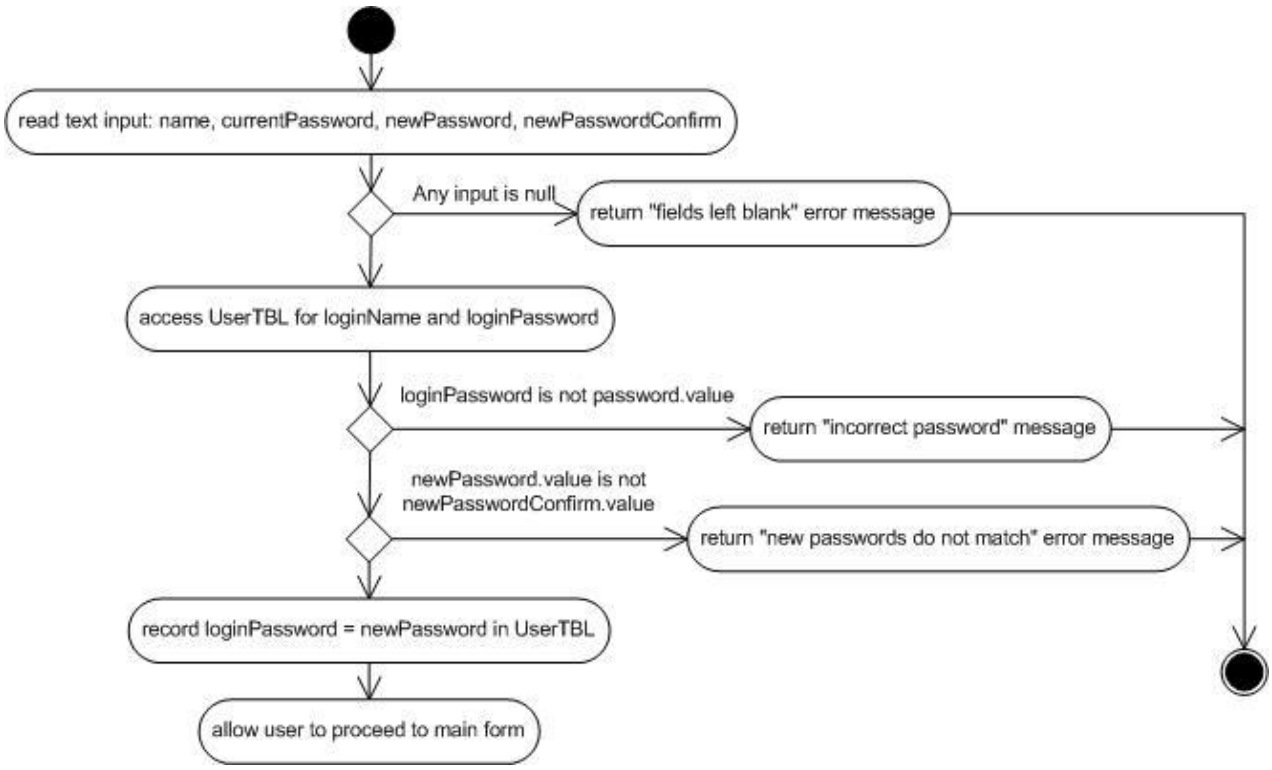
4.12.3 Component Processing Detail

4.12.3.1 Algorithmic Model

```
        In: Name
        In: CurrentPassword
        In: NewPassword
        In: NewPasswordConfirm
        Out: message of success or message of failure

Submit( In: Name, CurrentPassword, NewPassword, NewPasswordConfirm
        Out: message )
        validate inputs
        On Error: send message of failure and exit the routine
```

```
lookup from UserTBL (where Name.Value = LoginName) -> LoginName
lookup from UserTBL (where Password.Value = LoginPassword) -> LoginPassword
CorrectPassword = LoginPassword
if CorrectPassword is not Password.Value
        send message of failure and exit the routine
if NewPassword.Value is not NewPasswordConfirm.Value
        send message of failure and exit the routine
record to UserTBL -> LoginPassword = NewPassword.Value
if no errors occurred:
        send message of success
```

4.12.3.2 Activity diagram – ChangePassword( )


read text input: name, currentPassword, newPassword, newPasswordConfirm
Any input is null
return "fields left blank" error message
access UserTBL for loginName and loginPassword
loginPassword is not password.value
return "incorrect password" message
newPassword.value is not newPasswordConfirm.value
return "new passwords do not match" error message
record loginPassword = newPassword in UserTBL
allow user to proceed to main form

4.13 Description of Log Out

This component will allow the user to log out of the system.

4.13.1 Processing Narrative

A user wishes to log out of the system. First, the user must be on the main page of the tracking tool. The user must click on the button labeled *Log Off*. The user will be offically logged off of the tracking tool and must re−enter their username and password to use the tool again.

4.13.2 Interface Description

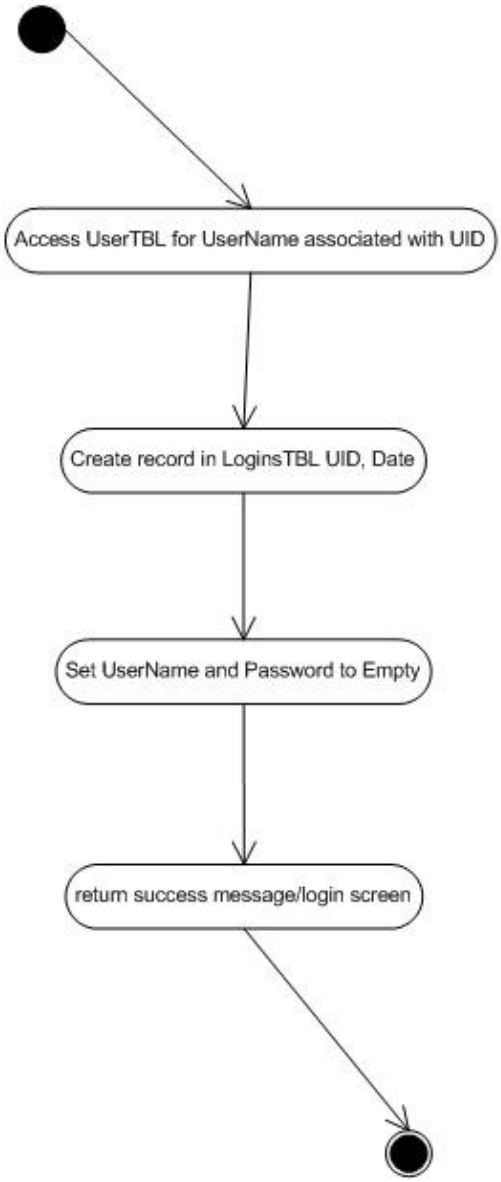
This component will be a button named *Log Off*. It will be located on the main page of the tracking tool.

4.13.3 Component Processing Detail

4.13.3.1 Algorithmic Model

```
        In: Name
        In: Password
        Out: Name
        Out: Password
        Out: Login Screen or message of failure

Submit( In: Name, Password
        Out: message and login Screen )
        On Error: send message of failure and exit the routine
        lookup  from UserTBL (where UserName.value=Name) ->uId
        ID=uId
        Add to LoginsTBL uId.Value = ID
        Add to LoginsTBL Date.Value = CurentDate

        Name=" "
        Password=" "
        if errors occurred:
                send message of failure and exit the routine
        if no errors occurred:
                 return screen to login page of tracking tool.
```

4.13.3.2 Activity diagram for the LogOut Module


Access UserTBL for UserName associated with UID
Create record in LoginsTBL UID, Date
Set UserName and Password to Empty
return success message/login screen

4.14 Description of Internal Link to an Object

This component will allow the user to link an internal object to the object type item.

4.14.1 Processing Narrative

A user wishes to link an internal object to an object type item. First, the user must be logged on to the tracking tool. On the Oject Type Item form, the user will click the *Add Internal Link* button. The user will be given a list of object type items stored in the database. The user will select an item and click the *OK* button.

4.14.2 Interface Description

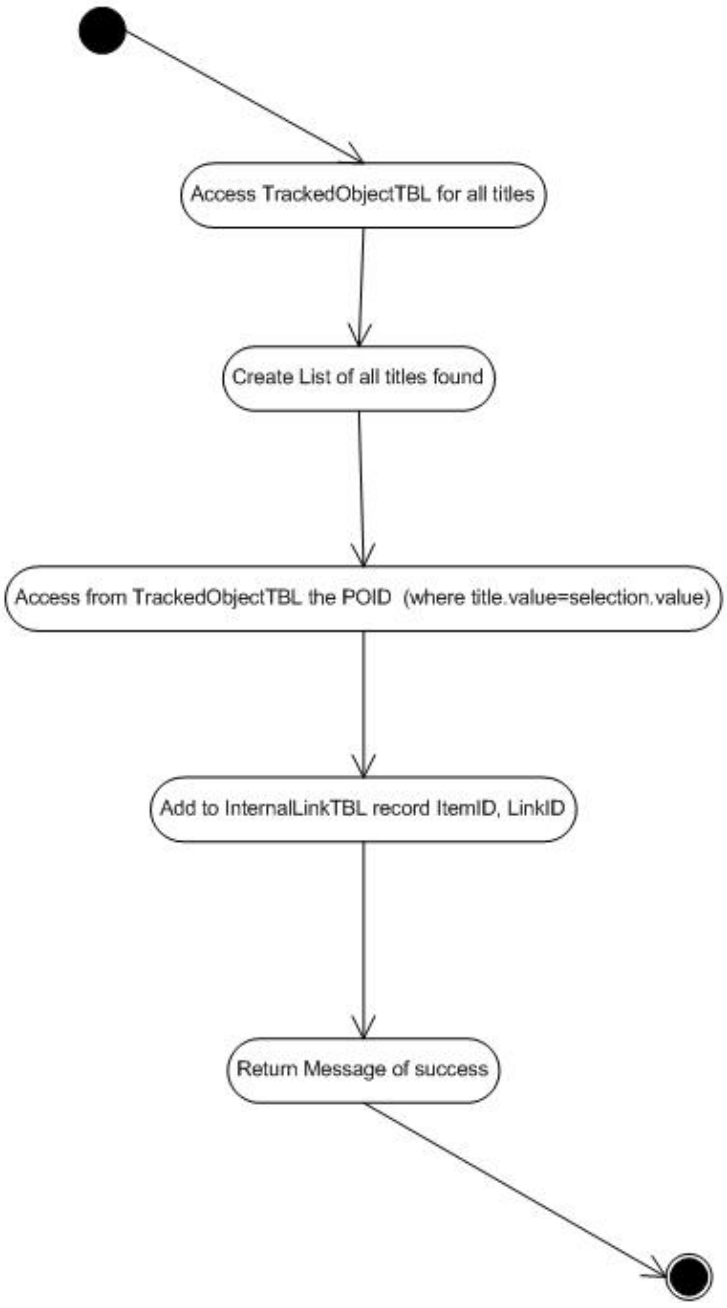
This component will be a button named *Add Internal Link* on the Item Screen.

4.14.3 Component Processing Detail

4.14.3.1 Algorithmic Model

```
        In: ItemID
        Out: message of success or failure


Submit (In: ItemID
        Out: message of success or failure
        On Error: send message of failure and exit the routine
        Populate a drop down list prompt with the following information:
                Title of all Items in TrackedObjectTBL

Ok      (In: Selection
         Out: LinkID)
         lookup from TrackedObjectTBL (where title.value=selection.value)->POID
         LinkID=POID
         return LinkID

        Add to InternalLinkTBL record ItemID, LinkID
        if errors occurred:
                send message of failure and exit the routine
        if no errors occurred:
                send message of success and return to the item screen.
```

4.14.3.2 Activity diagram for the Internal Link Module


Access TrackedObjectTBL for all titles
Create List of all titles found
Access from TrackedObjectTBL the POID (where title.value=selection.value)
Add to InternalLinkTBL record ItemID, LinkID
Return Message of success

4.15 Description of External Link to an Object

This component will allow the user to link an external object to the object type item.

4.15.1 Processing Narrative

A user wishes to link an external object to an object type item. First, the user must be logged on to the tracking tool. On the Oject Type Item form, the user will click the *Add External Link* button. The user will be given a list of entries to link to the item. The user will select an entry and click the *OK* button.

4.15.2 Interface Description

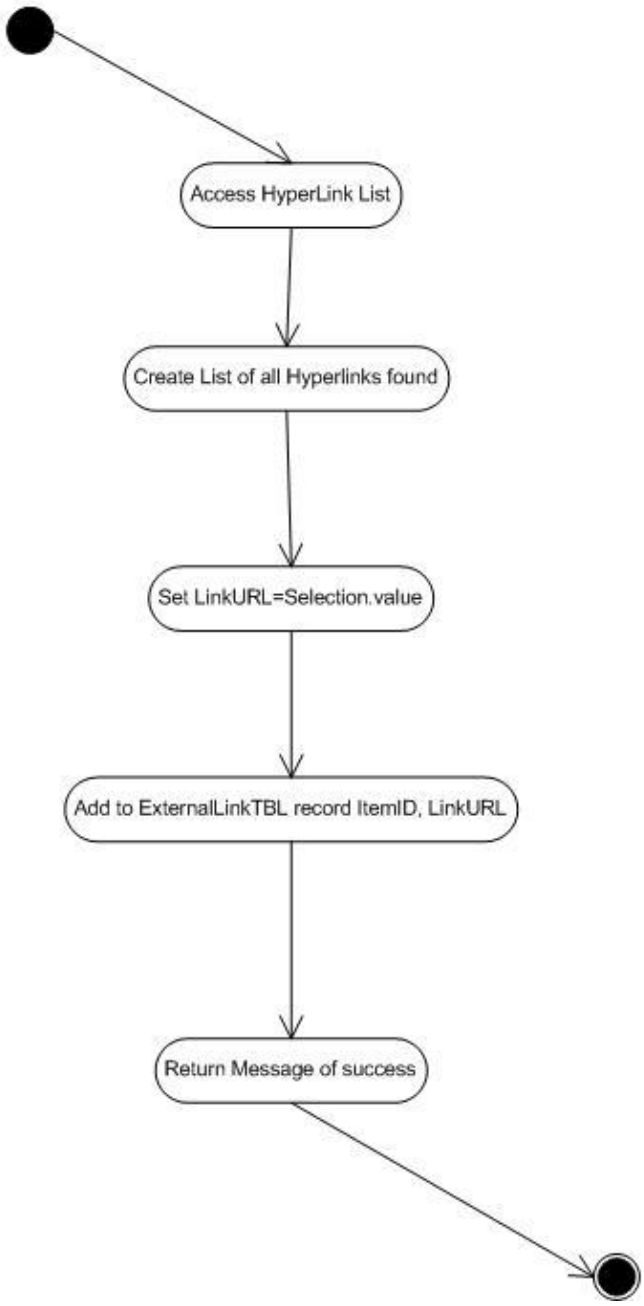
This component will be a button named *Add External Link* on the Item Screen.

4.15.3 Component Processing Detail

4.15.3.1 Algorithmic Model

```
        In: ItemID
        Out: message of success or failure


Submit (In: ItemID
        Out: message of success or failure
        On Error: send message of failure and exit the routine
        Populate a drop down menu prompt with the following information:
                HyperLink Items

Ok      (In: Selection
         Out: LinkURL)
         LinkURL=Selection.value
         return LinkURL

        Add to ExternalLinkTBL record ItemID, LinkURL
        if errors occurred:
                send message of failure and exit the routine
        if no errors occurred:
                send message of success and return to the item screen.
```

4.15.3.2 Activity diagram for the External Link Module


Access HyperLink List
Create List of all Hyperlinks found
Set LinkURL=Selection.value
Add to ExternalLinkTBL record ItemID, LinkURL
Return Message of success

4.16 Description of Add Attachement to an Item

This component will allow the user to attach a file to the object type item.

4.16.1 Processing Narrative

A user wishes to attach a file to an object type item. First, the user must be logged on to the tracking tool. On the Oject Type Item form, the user will click the *Add Attachment* button. The user will be given a list of files to attach to the item. The user will select a file and click the *OK* button.

4.16.2 Interface Description

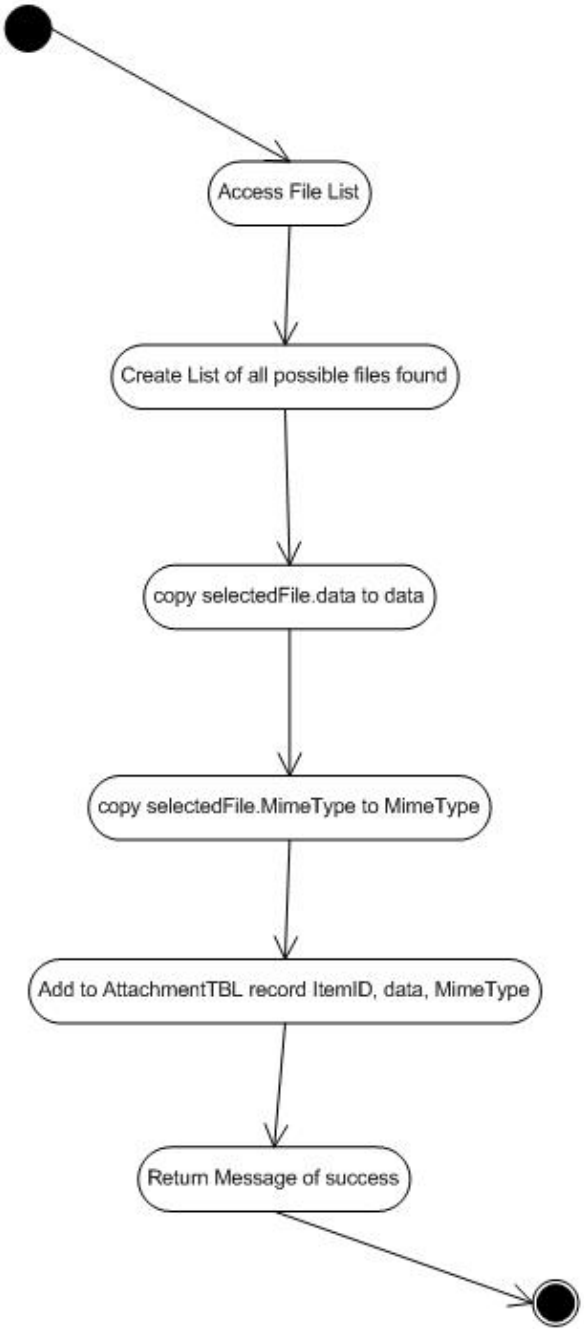
This component will be a button named *Add Attachment* on the Item Screen.

4.16.3 Component Processing Detail

4.16.3.1 Algorithmic Model

```
        In: ItemID
        Out: message of success or failure


Submit (In: ItemID
        Out: message of success or failure
        On Error: send message of failure and exit the routine
        Populate a drop down list prompt with the following information:
                File Names
Ok      (In: SelectedFile
         Out: data
         Out: MimeType)
         copy selectedFile.data to data
         copy selectedFile.MimeType to MimeType
         return data and MimeType

        Add to AttachmentTBL record ItemID, data, MimeType
        if errors occurred:
                send message of failure and exit the routine
        if no errors occurred:
                send message of success and return to the item screen.
```

4.16.3.2 Activity diagram for the Attachment Module


Access File List
Create List of all possible files found
copy selectedFile.data to data
copy selectedFile.MimeType to MimeType
Add to AttachmentTBL record ItemID, data, MimeType
Return Message of success

4.17 Description of Change Log for Watch Item

This component allows the user to view all changes made to an Watch item in the tracking tool.

4.17.1 Processing Narrative

First, the user must be logged into the tracking tool. The user must click on the tab labeled *Watch*. The user sees a list of items. The user hightlights an item and clicks on the *View* button. The user will then see the Item Screen. On this screen will be a button called *Change Log*. Once the user clicks on the button, the user can view changes made to the item.

4.17.2 Interface Description

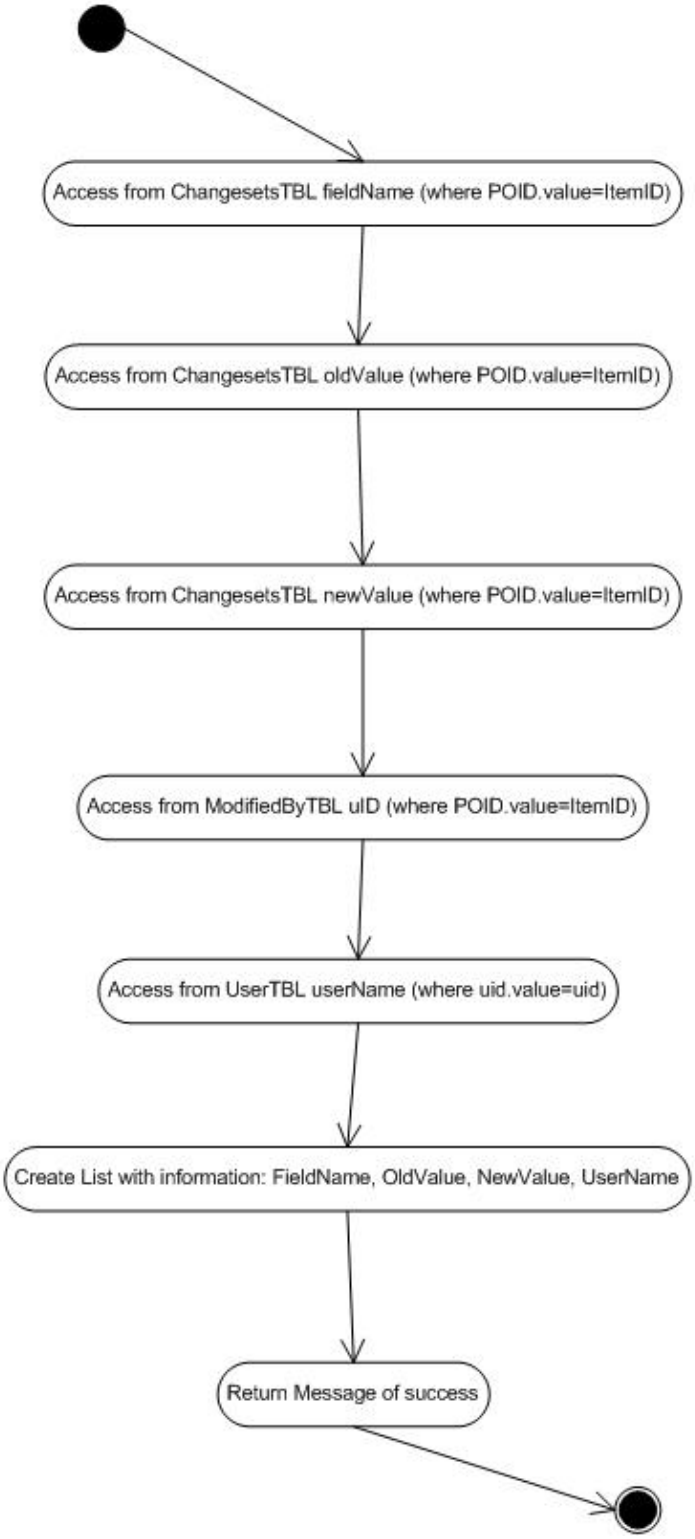
This component will be a button named *Change Log*. It will be located on the Watch Item screen of the tracking tool.

4.17.3 Component Processing Detail

4.17.3.1 Algorithmic Model

```
        In: ItemID
        Out: UserName
        Out: FieldName
        Out: OldValue
        Out: NewValue
        Out:  message of failure or list of changes made to the item

Submit( In: ItemID
        Out: UserName
        Out: FieldName
        Out: OldValue
        Out: NewValue
        Out: LogInDate
        Out: list of changes made to the item  )
        On Error: send message of failure and exit the routine
        lookup  from ChangesetsTBL (where POID.value=ItemID) ->fieldname
        lookup  from ChangesetsTBL (where POID.value=ItemID) ->oldvalue
        lookup  from ChangesetsTBL (where POID.value=ItemID) ->newvalue
        lookup  from ModifiedByTBL (where POID.value=ItemID) ->uid
        lookup  from UserTBL (where uid.value=uid) ->username
        FieldName=fieldname
        OldValue=oldvalue
        NewValue=newvalue
        UserName=username
        Populate List with the following information:
                FieldName, OldValue, NewValue, UserName

        if errors occurred:
                send message of failure and exit the routine
        if no errors occurred:
                 return screen to login page of tracking tool.
```

4.17.3.2 Activity diagram for the Change Log for a Watch Item Module


Access from ChangesetsTBL fieldName (where POID.value=ItemID)
Access from ChangesetsTBL oldValue (where POID.value=ItemID)
Access from ChangesetsTBL newValue (where POID.value=ItemID)
Access from ModifiedByTBL uID (where POID.value=ItemID)
Access from UserTBL userName (where uid.value=uid)
Create List with information: FieldName, OldValue, NewValue, UserName
Return Message of success

4.18 Description of Change Log for Inventory Item

This component allows the user to view all changes made to an Inventory item in the tracking tool.

4.18.1 Processing Narrative

First, the user must be logged into the tracking tool. The user must click on the tab labeled *Inventory*. The user sees a list of items. The user highlights an item and clicks on the *View* button. The user will then see the Item Screen. On this screen will be a button called *Change Log*. Once the user clicks on the button, the user can view changes made to the item.

4.18.2 Interface Description

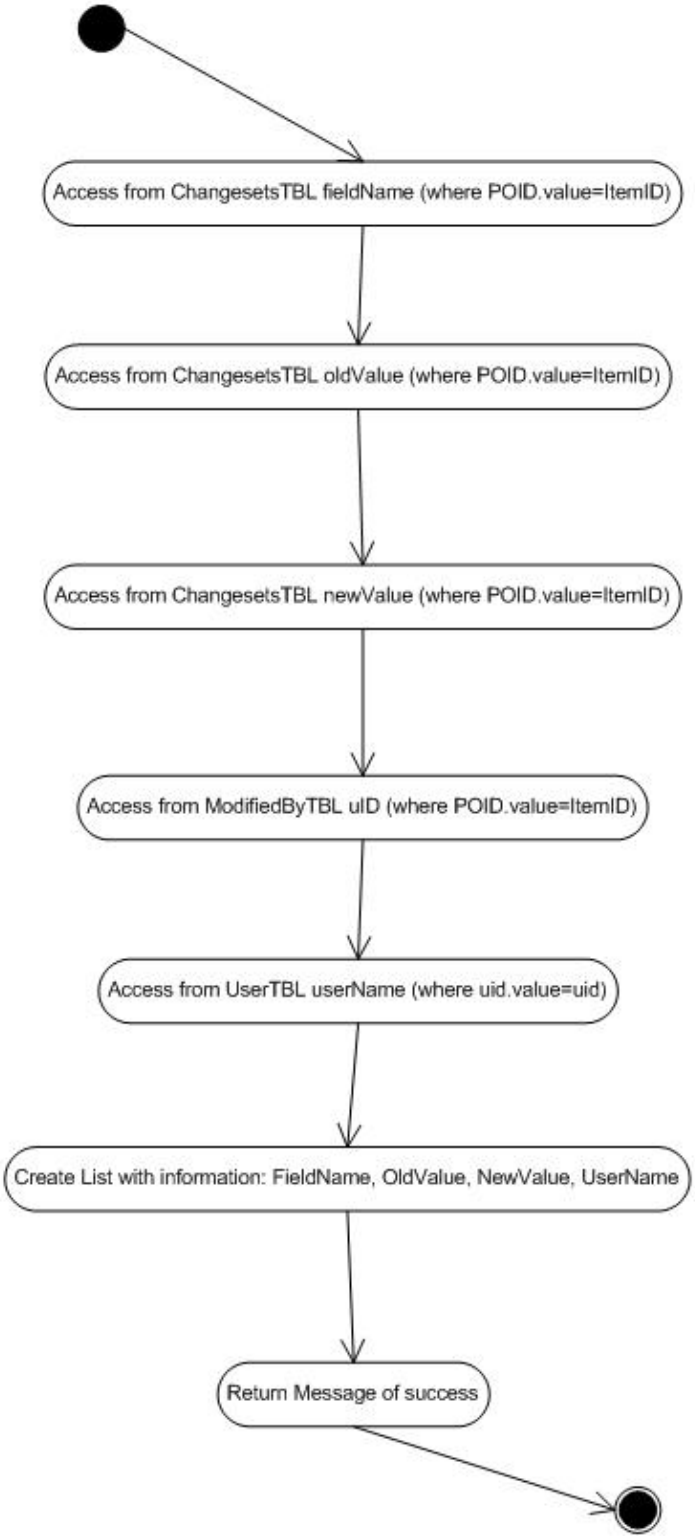
This component will be a button named *Change Log*. It will be located on the Inventory Item screen of the tracking tool.

4.18.3 Component Processing Detail

4.18.3.1 Algorithmic Model

```
        In: ItemID
        Out: UserName
        Out: FieldName
        Out: OldValue
        Out: NewValue
        Out:  message of failure or list of changes made to the item

Submit( In: ItemID
        Out: UserName
        Out: FieldName
        Out: OldValue
        Out: NewValue
        Out: LogInDate
        Out: list of changes made to the item  )
        On Error: send message of failure and exit the routine
        lookup  from ChangesetsTBL (where POID.value=ItemID) ->fieldname
        lookup  from ChangesetsTBL (where POID.value=ItemID) ->oldvalue
        lookup  from ChangesetsTBL (where POID.value=ItemID) ->newvalue
        lookup  from ModifiedByTBL (where POID.value=ItemID) ->uid
        lookup  from UserTBL (where uid.value=uid) ->username
        FieldName=fieldname
        OldValue=oldvalue
        NewValue=newvalue
        UserName=username
        Populate List with the following information:
                FieldName, OldValue, NewValue, UserName

        if errors occurred:
                send message of failure and exit the routine
        if no errors occurred:
                 return screen to login page of tracking tool.
```

4.18.3.2 Activity diagram for the Change Log for an Inventory Item Module


Access from ChangesetsTBL fieldName (where POID.value=ItemID)
Access from ChangesetsTBL oldValue (where POID.value=ItemID)
Access from ChangesetsTBL newValue (where POID.value=ItemID)
Access from ModifiedByTBL uID (where POID.value=ItemID)
Access from UserTBL userName (where uid.value=uid)
Create List with information: FieldName, OldValue, NewValue, UserName
Return Message of success

4.19 Description of Email to the Assigned To Person of Watch List Item

This component will allow the user to email a watch list item and details to a user.

4.19.1 Processing Narrative

A user wishes to email to the person assigned to a watch list item. First, the user must be logged on to the tracking tool. On the Watch List Item form, the user will click the *Email* button. The user will be given a list of names of people on the team. The user will select a name and click the *OK* button.

4.19.2 Interface Description

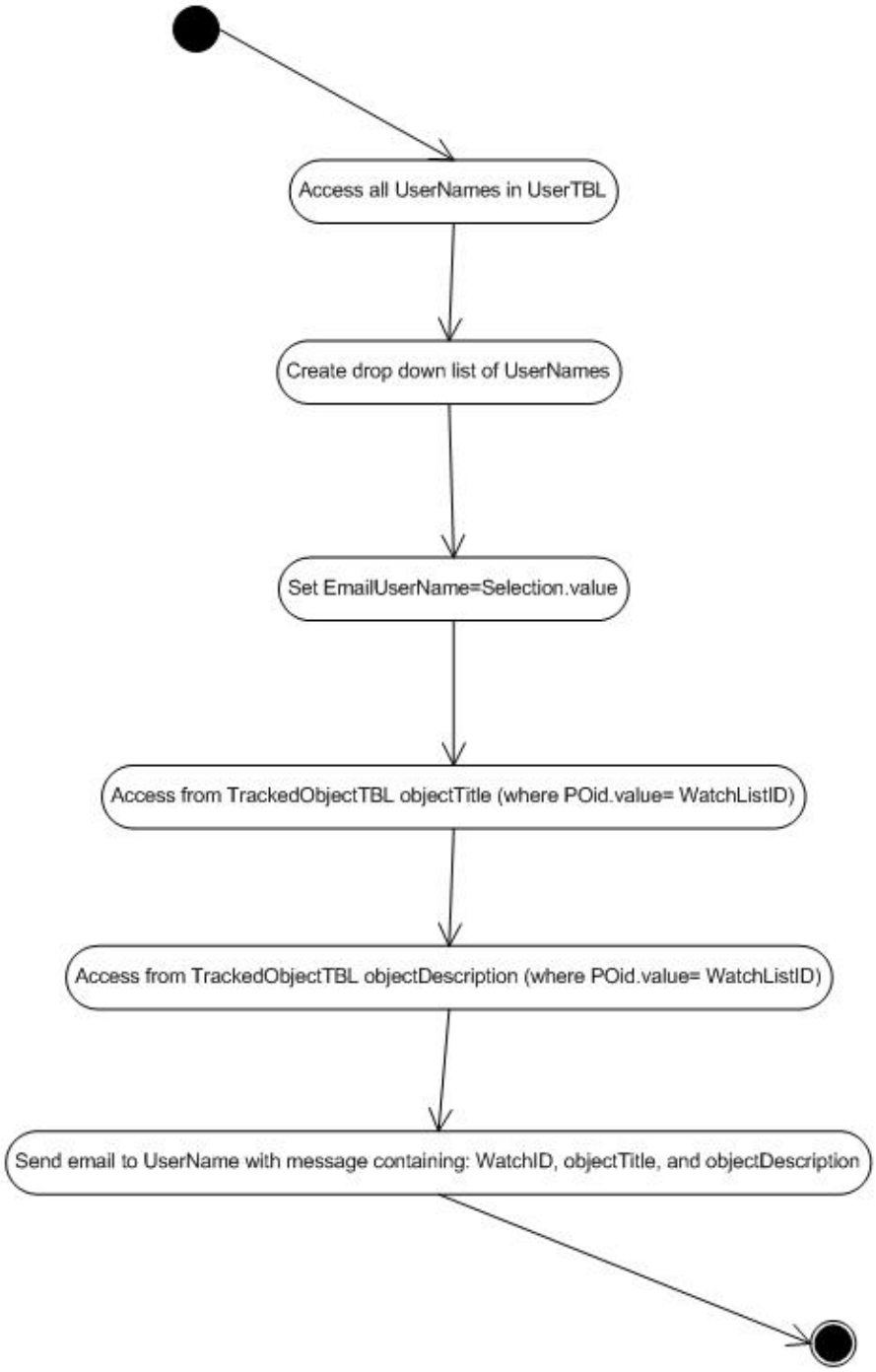
This component will be a button named *Email* on the Watch Item Screen.

4.19.3 Component Processing Detail

4.19.3.1 Algorithmic Model

```
        In: WatchListID
        In: Name
        Out: Name
        Out: WatchListID
        Out: Title
        Out Description
        Out: Email or message of failure


Submit (In: WatchListID
        Out:WatchListID
        Out: Name
        Out: WatchListID
        Out: Title
        Out Description
        Out: message and email)
        On Error: send message of failure and exit the routine

        Populate a drop down list prompt with the following information:
                UserNames of all Items in UserTBL
Ok      (In: Selection
         Out: Name)
        Name=Selection.value
         return Name


        lookup  from TrackedObjectTBL (where POid.value= WatchListID) ->ObjectTitle
        lookup  from TrackedObjectTBL (where POid.value= WatchListID) ->ObjectDescription
        Title=OjectTitle
        Description=ObjectDescription
        message=("Dear %n, \r Watch List Item %n with Title %n is assigned to you.  \r The descri
        if errors occurred:
                send message of failure and exit the routine
        if no errors occurred:
                send email to Name with message.
```

4.19.3.2 Activity diagram for Email to Assigned To Person of a Watch Item Module


Access all UserNames in UserTBL
Create drop down list of UserNames
Set EmailUserName=Selection.value
Access from TrackedObjectTBL objectTitle (where POid.value= WatchListID)
Access from TrackedObjectTBL objectDescription (where POid.value= WatchListID)
Send email to UserName with message containing: WatchID, objectTitle, and objectDescription

4.20 Description of Email to the Assigned To Person of Inventory Item

This component will allow the user to email an Inventory item and details to a user.

4.20.1 Processing Narrative

A user wishes to email to the person assigned to an Inventory item. First, the user must be logged on to the tracking tool. On the Inventory Item form, the user will click the *Email* button. The user will be given a list of names of people on the team. The user will select a name and click the *OK* button.

4.20.2 Interface Description

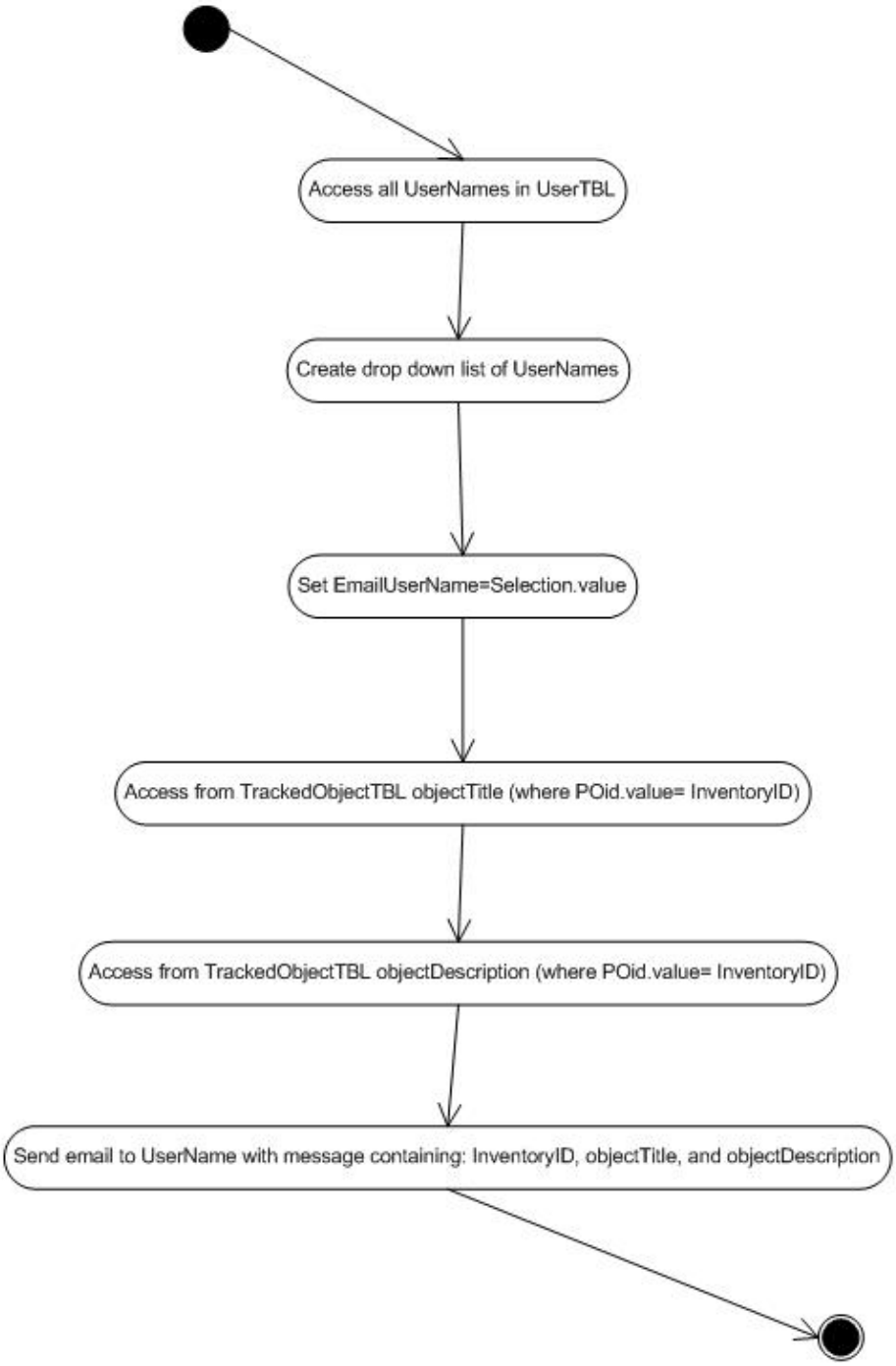
This component will be a button named *Email* on the Inventory Item Screen.

4.20.3 Component Processing Detail

4.20.3.1 Algorithmic Model

```
        In: InventoryID
        In: Name
        Out: Name
        Out: InventoryID
        Out: Title
        Out Description
        Out: Email or message of failure


Submit (In: InventoryID
        Out:InventorytID
        Out: Name
        Out: Title
        Out Description
        Out: message and email)
        On Error: send message of failure and exit the routine
        Populate a drop down list prompt with the following information:
                UserNames of all Items in UserTBL
Ok      (In: Selection
         Out: Name)
        Name=Selection.value
         return Name

        lookup  from TrackedObjectTBL (where POid.value= InventoryID) ->ObjectTitle
        lookup  from TrackedObjectTBL (where POid.value= InventoryID) ->ObjectDescription
        Title=OjectTitle
        Description=ObjectDescription
        message=("Dear %n, \r Inventory Item %n with Title %n is assigned to you.  \r The descrip
        if errors occurred:
                send message of failure and exit the routine
        if no errors occurred:
                send email to Name with message.
```

4.20.3.2 Activity diagram for Email to Assigned To Person of an Inventory Item Module


Access all UserNames in UserTBL
Create drop down list of UserNames
Set EmailUserName=Selection.value
Access from TrackedObjectTBL objectTitle (where POid.value= InventoryID)
Access from TrackedObjectTBL objectDescription (where POid.value= InventoryID)
Send email to UserName with message containing: InventoryID, objectTitle, and objectDescription

4.21 Description of View Inventory List

This component will allow a user to view the a list of all Inventory Items that are currently in the system.

4.21.1 Processing Narrative

A user wishes to view the list of Inventory Items currently in the database. In order to do this, the user has to have been authenticated and authorized. Next, the user will click on the *Inventory* Tab. A list will then be given to the user of all the Inventory Items.

4.21.2 Interface Description

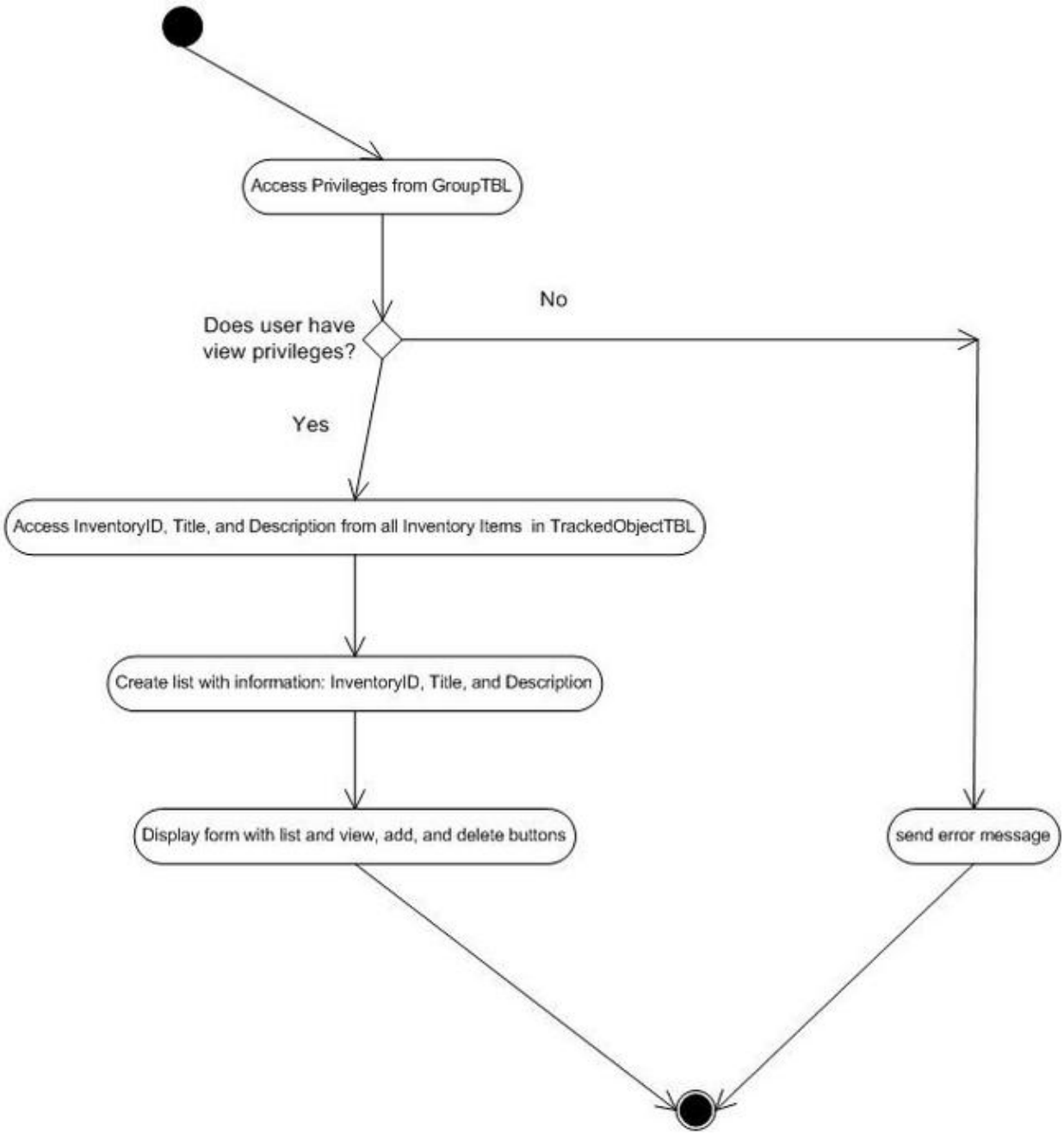
On the *Inventory* tab, a list will be given of the Inventory Items.

4.21.3 Component Processing Detail

4.21.3.1 Algorithmic Model

```
        In: ObjectType
        Out: form that contains a list of all items of the Inventory Type in the database and act


ViewInventoryList( In: ObjectType )
        OnError: send message of failure and exit routine
        Populate a list with the following information:
                Name, Title, and Description for all type.value=ObjectType in TrackedObjectTBL


ViewInvetoryDetails( In: POID from selection )
        This will direct the user to the View Inventory object module

Modify( in: POID from selection )
        This algorithm is described in the Modify Inventory object component description.

Delete( in: POID from selection )
        This algorithm is described in the Delete Inventory object component description.

Add( in: POID from selection )
        This algorithm is described in the Add Inventory object component description.
```

4.21.3.2 Activity diagram for View Inventory List Module


Access Privileges from GroupTBL
Does user have view privileges?
No
Yes
Access InventoryID, Title, and Description from all Inventory Items in TrackedObjectTBL
Create list with information: InventoryID, Title, and Description
Display form with list and view, add, and delete buttons
send error message

4.22 Description of View Watch List

This component will allow a user to view the a list of all Watch Items that are currently in the system.

4.22.1 Processing Narrative

A user wishes to view the list of Watch Items currently in the database. In order to do this, the user has to have been authenticated and authorized. Next, the user will click on the *Watch* Tab. A list will then be given to the user of all the Watch Items.

4.22.2 Interface Description

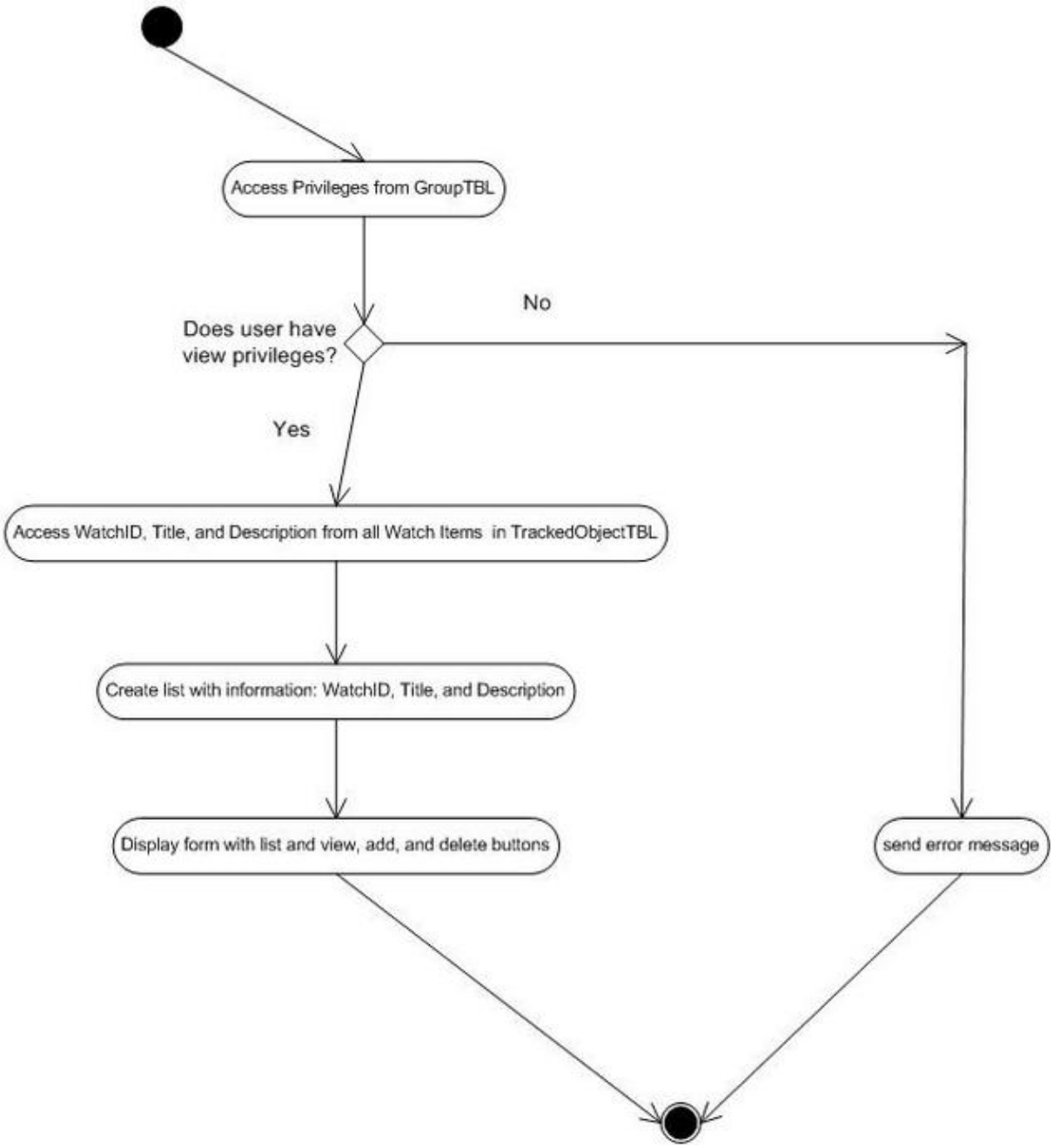
On the *Watch* tab, a list will be given of the Inventory Items.

4.22.3 Component Processing Detail

4.22.3.1 Algorithmic Model

```
        In: ObjectType
        Out: form that contains a list of all items of the Watch Type in the database and actions


ViewWatchList( In: ObjectType )
        OnError: send message of failure and exit routine
        Populate a list with the following information:
                Name, Title, and Description for all type.value=ObjectType in TrackedObjectTBL


ViewWatchDetails( In: POID from selection )
        This will direct the user to the View Watch object module

Modify( in: POID from selection )
        This algorithm is described in the Modify Watch object component description.

Delete( in: POID from selection )
        This algorithm is described in the Delete Watch object component description.

Add( in: POID from selection )
        This algorithm is described in the Add Watch object component description.
```

4.22.3.2 Activity diagram for View Watch List Module


Access Privileges from GroupTBL
Does user have view privileges?
No
Yes
Access WatchID, Title, and Description from all Watch Items in TrackedObjectTBL
Create list with information: WatchID, Title, and Description
Display form with list and view, add, and delete buttons
send error message

4.23 Description of View Watch Object

This component will allow a user to view the details of a Watch Item that is currently in the system.

4.23.1 Processing Narrative

A user wishes to view the details of a Watch Item currently in the database. In order to do this, the user has to have been authenticated and authorized. Next, the user will click on the *Watch* Tab. A list will then be given to the user of all the Watch Items. The user will then highlight an item to view and click on the *View* button.

4.23.2 Interface Description

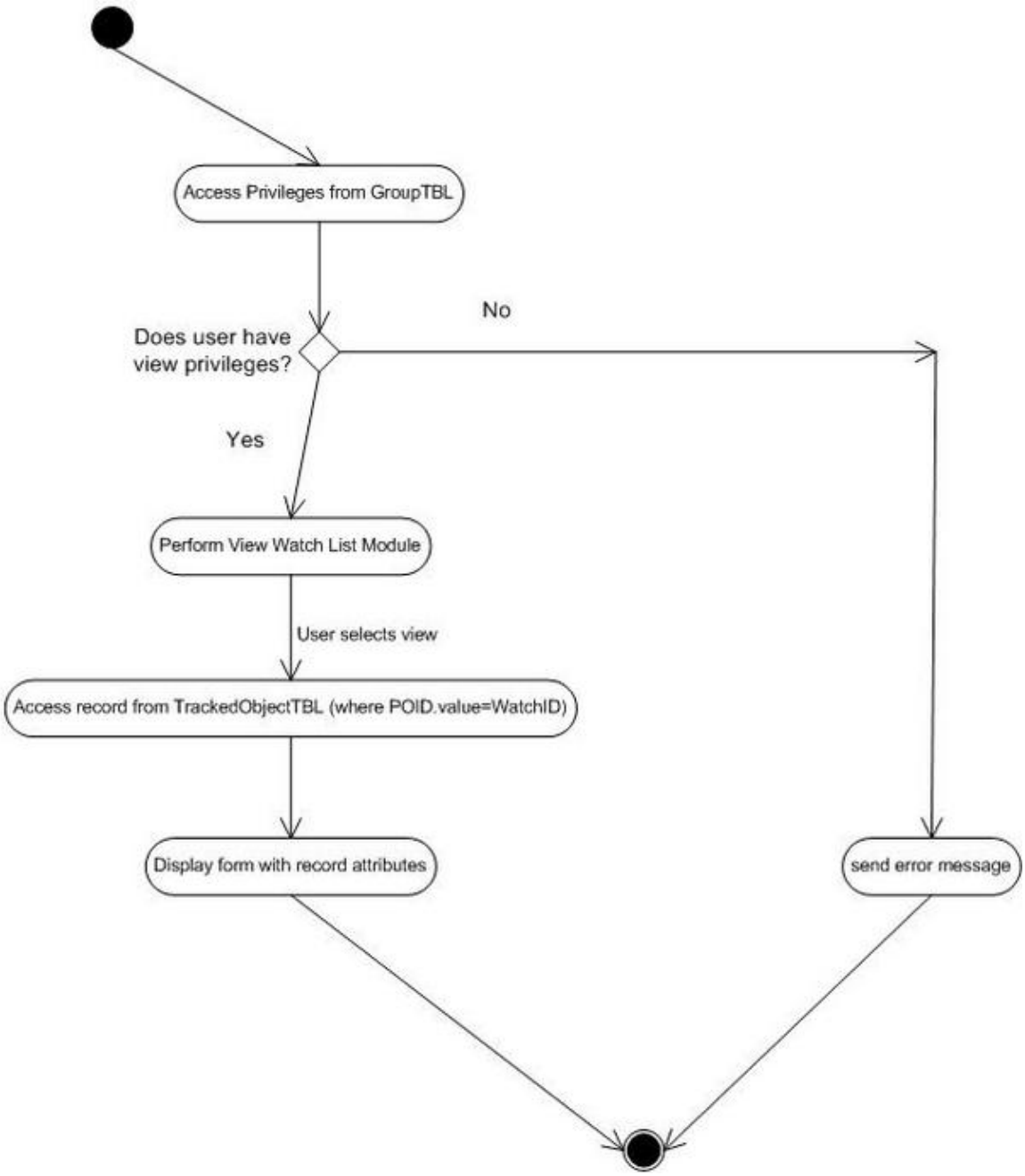
On the *Watch* tab, a list will be given of the Watch Items. The user will click on an item to highlight it and then click on the *View* button.

4.23.3 Component Processing Detail

4.23.3.1 Algorithmic Model

```
        In: WatchID from selection
        Out: form that contains the details of the Watch Item.

ViewWatchList (ObjectType)
        This algorithm is described in the View Watch List component description.

Submit [(In:WatchID from selection
        Out: Form with Watch Object details
        On Error: send message of failure and exit the routine

ViewWatchItem [(In: WatchID
                Out: ViewWatchItem Form
                lookup record from TrackedObjectTBL (where POID.value=WatchID)
                Fill in ViewWatchItem form with record attributes
              ]
        if errors occurred:
                send message of failure and exit the routine
        if no errors occurred:
                send message of success and return to the item screen.
        ]
```

4.23.3.2 Activity diagram for View Watch Item Module



4.24 Description of View Inventory Object

This component will allow a user to view the details of an Inventory Item that is currently in the system.

4.24.1 Processing Narrative

A user wishes to view the details of an Inventory Item currently in the database. In order to do this, the user has to have been authenticated and authorized. Next, the user will click on the *Inventory* Tab. A list will then be given to the user of all the Inventory Items. The user will then highlight an item to view and click on the *View* button.

4.24.2 Interface Description

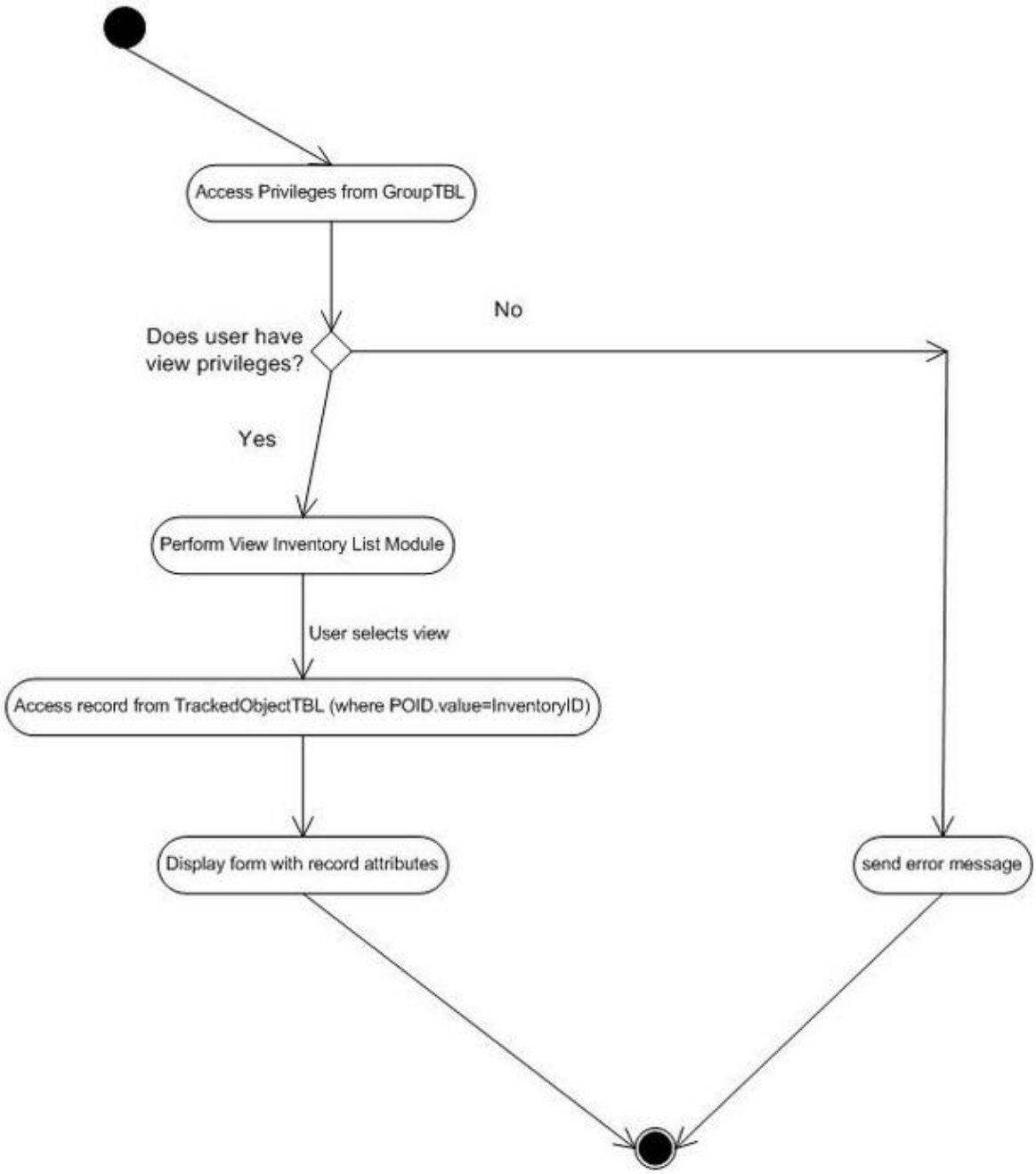
On the *Inventory* tab, a list will be given of the Inventory Items. The user will click on an item to highlight it and then click on the *View* button.

4.24.3 Component Processing Detail

4.24.3.1 Algorithmic Model

```
        In: InventoryID from selection
        Out: form that contains the details of the Inventory Item.


ViewInventoryList (ObjectType)
        This algorithm is described in the View Inventory List component description.

Submit [(In: InventoryID from selection
        Out: Form with Inventory Object details
        On Error: send message of failure and exit the routine

ViewInventoryItem [(In: InventoryID
                Out: ViewInventoryItem Form
                lookup record from TrackedObjectTBL (where POID.value=InventoryID)
                Fill in ViewInventoryItem form with record attributes
              ]
        if errors occurred:
                send message of failure and exit the routine
        if no errors occurred:
                send message of success and return to the item screen.
        ]
```

4.24.3.2 Activity diagram for View Inventory Item Module



4.25 Description of Add Watch Object

This component will allow a user to add a watch list object to the watch list, and describe the attributes of that object.

4.25.1 Processing Narrative

A user wishes to add a watch to the watch list. The user must be authenticated and authorized to do so. To get to the watches window, the user must select the *watches* tab on the main screen. There will be a command button called *add watch*. This will take the user to the add watch interface, described below.

4.25.2 Interface Description

On this form, there will be both required and non−required fields. To start out, the watch list objects will have attributes that are considered necessary, and attributes that are common to both watch objects and inventory objects. Other attributes may be added by an authorized user in the configuration section.

The required elements on this form will be described as follows. There will be a dropdown list to choose the type of this object (for example: QD, BR, MP, RP, etc). There will be text boxes for the following things: the create date, a short description (or title), a long description, and the originator. There will be a dropdown list of users (populated from the user table in the database) to assign this item to.

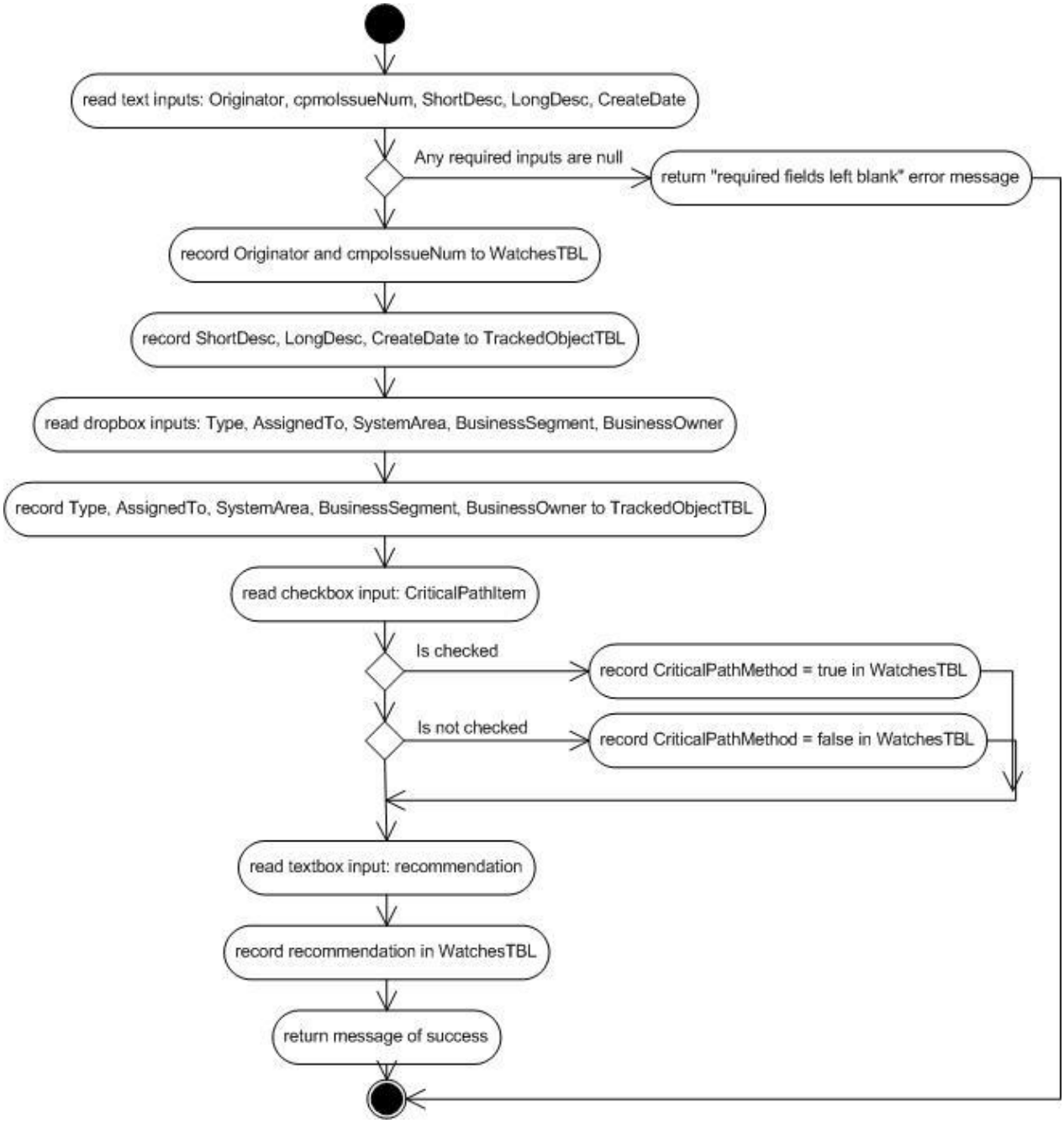
The non−required elements on this form will be described as follows. There will be a text box to enter the cPMO issue number. There will be a checkbox control to indicate whether or not this object is a critical path (or hot) item. There will be dropdown boxes for the following things: the system area (populated from the system area table), the business segment (populated from the business segments table), and the business owner. There will be a recommendations text box. There will also be controls to add attachments and to link files. (Those components are described elsewhere.) There will be a *cancel* command button that will enable the user to cancel the process of adding a watch item at any time. There will be a *save* button that will save the changes to the database.

4.25.3 Component Processing Detail

4.25.3.1 Algorithmic Model

```
        In: Originator, CreateDate, ShortDesc, LongDesc
        In: Type, AssignedTo
        In: CriticalPathItem = true, false
        In: cpmoIssueNum, Recommendation
        In: SystemArea, BusinessSegment, BusinessOwner
        Out: message of success or message of failure

Save( In: All from above
      Out: message )
        validate inputs
        On Error: send message of failure and exit the routine
        record to WatchesTBL -> Originator, cmpoIssueNum
        record to TrackedObjectTBL -> ShortDesc, LongDesc, CreateDate
        record to TrackedObjectTBL -> Type, AssignedTo, SystemArea, BusinessSegment, BusinessOwne
        record to WatchesTBL -> CriticalPathItem
        record to WatchesTBL -> recommendation
        if no errors occurred:
                send message of success
```

4.25.3.2 Activity diagram for Add Watch Module


read text inputs: Originator, cpmoIssueNum, ShortDesc, LongDesc, CreateDate
Any required inputs are null
return "required fields left blank" error message
record Originator and cmpoIssueNum to WatchesTBL
record ShortDesc, LongDesc, CreateDate to TrackedObjectTBL
read dropbox inputs: Type, AssignedTo, SystemArea, BusinessSegment, BusinessOwner
record Type, AssignedTo, SystemArea, BusinessSegment, BusinessOwner to TrackedObjectTBL
read checkbox input: CriticalPathItem
Is checked
record CriticalPathMethod = true in WatchesTBL
Is not checked
record CriticalPathMethod = false in WatchesTBL
read textbox input: recommendation
record recommendation in WatchesTBL
return message of success

4.26 Description of Add Inventory Object

This component will allow a user to add an inventory object to the inventory list, and describe the attributes of that object.

4.26.1 Processing Narrative

A user wishes to add an inventory object to the inventory list. The user must be authenticated and authorized to do so. To get to the inventory window, the user must select the *inventory* tab on the main screen. There will

be a command button called *add inventory*. This will take the user to the add inventory interface, described below.

4.26.2 Interface Description

On this form, there will be both required and non−required fields. To start out, the inventory objects will have attributes that are considered necessary, and attributes that are common to both inventory objects and watch objects. Other attributes may be added by an authorized user in the configuration section.

The required elements on this form will be described as follows. There will be text boxes for the following things: the date created, a short description (or title), and a long description. There will be a dropdown list of users (populated from the user table) to select who this object is assigned to. There will also be a dropdown list to select this object's type (detailed design, conceptual design, etc).

The non−required elements on this form will be described as follows. There will be a text boxes to enter the release, the planned draft date, the actual draft date, the planned internal review date, the actual internal review date, the planned business review date, the actual business review date, the planned quality review date, and the actual quality review date. There will be dropdown lists of the name of the team (populated from the team table), the business segment, the system area, and the business owner. There will also be controls to add application systems, and technologies used. There will also be controls to add attachments and to link files. (Those components are described elsewhere.) There will be a *cancel* command button that will enable the user to cancel the process of adding an inventory item at any time. There will be a *save* button that will save the changes to the database.

4.26.3 Component Processing Detail

4.26.3.1 Algorithmic Model

```
        In: ShortDesc, LongDesc, CreateDate
        In: Type, AssignedTo
        In: Release, PlannedDraftDate, ActualDraftDate, PlannedInternalReviewDate,
            ActualInternalReviewDate, PlannedBusinessReviewDate, ActualBusinessReviewDate,
            PlannedQualityReviewDate, ActualQualityReviewDate
        In: Team, BusinessSegment, SystemArea, BusinessOwner
        In: ApplicationSystem, TechnologyUsed
        Out: message of success or message of failure

Save( In: All from above
      Out: message )
        validate inputs
        On Error: send message of failure and exit the routine
        record to TrackedObjectTBL -> DateCreated, ShortDesc, LongDesc
        record to DesignTBL -> Release, PlannedDraftDate, ActualDraftDate,
                    PlannedInternalReviewDate, ActualInternalReviewDate,
                    PlannedBusinessReviewDate, ActualBusinessReviewDate,
                    PlannedQualityReviewDate, ActualQualityReviewDate
        record to TrackedObjectTBL -> Type, AssignedTo, Team, BusinessSegment,
                    SystemArea, BusinessOwner
        record to DesignTBL -> ApplicationSystem, TechnologyUsed
                if no errors occurred:
                send message of success
```

4.26.3.2 Activity diagram for Add Inventory Module

read text inputs: ShortDesc, LongDesc, CreateDate
read text inputs: Release, pDraft, pInternalReview, pBusinessReview, pActualReview
read text inputs: aDraft, aInternalReview, aBusinessReview, aActualReview
Any required inputs are null
return "required fields left blank" error message
record ShortDesc, LongDesc, CreateDate to TrackedObjectTBL
record Release, pDraft, pInternalReview, pBusinessReview, pActualReview to DesignTBL
record aDraft, aInternalReview, aBusinessReview, aActualReview to DesignTBL
read dropbox inputs: Type, AssignedTo, SystemArea, BusinessSegment, BusinessOwner, Team
record Type, AssignedTo, SystemArea, BusinessSegment, BusinessOwner, and Team to TrackedObjectTBL
read list inputs: ApplicationSystem, TechnologyUsed
record ApplicationSystem and TechnologyUsed in DesignTBL
return message of success

4.27 Description of Modify Watch Object

This component will allow a user to modify a watch list object in the watch list.

4.27.1 Processing Narrative

A user wishes to modify a watch list item. The user must be authenticated and authorized to do so. To get to the watches window, the user must select the *watches* tab on the main screen. There will be a command button called *modify watch*. This will take the user to the modify watch interface, described below.

4.27.2 Interface Description

On this form, there will be both required and non−required fields. The required elements on this form will be described as follows. There will be a dropdown list to choose the type of this object (for example: QD, BR, MP, RP, etc). There will be text boxes for the following things: the create date, a short description (or title), a long description, and the originator. There will be a dropdown list of users (populated from the user table in the database) to assign this item to. All of these things will be editable.

The non−required elements on this form will be described as follows. There will be a text box to enter the cPMO issue number. There will be a checkbox control to indicate whether or not this object is a critical path (or hot) item. There will be dropdown boxes for the following things: the system area (populated from the system area table), the business segment (populated from the business segments table), and the business owner. There will be a recommendations text box. All of these things are editable. There will be a *cancel* command button that will enable the user to cancel the process of modifying a watch item at any time. There will be a *save* button that will save the changes to the database.

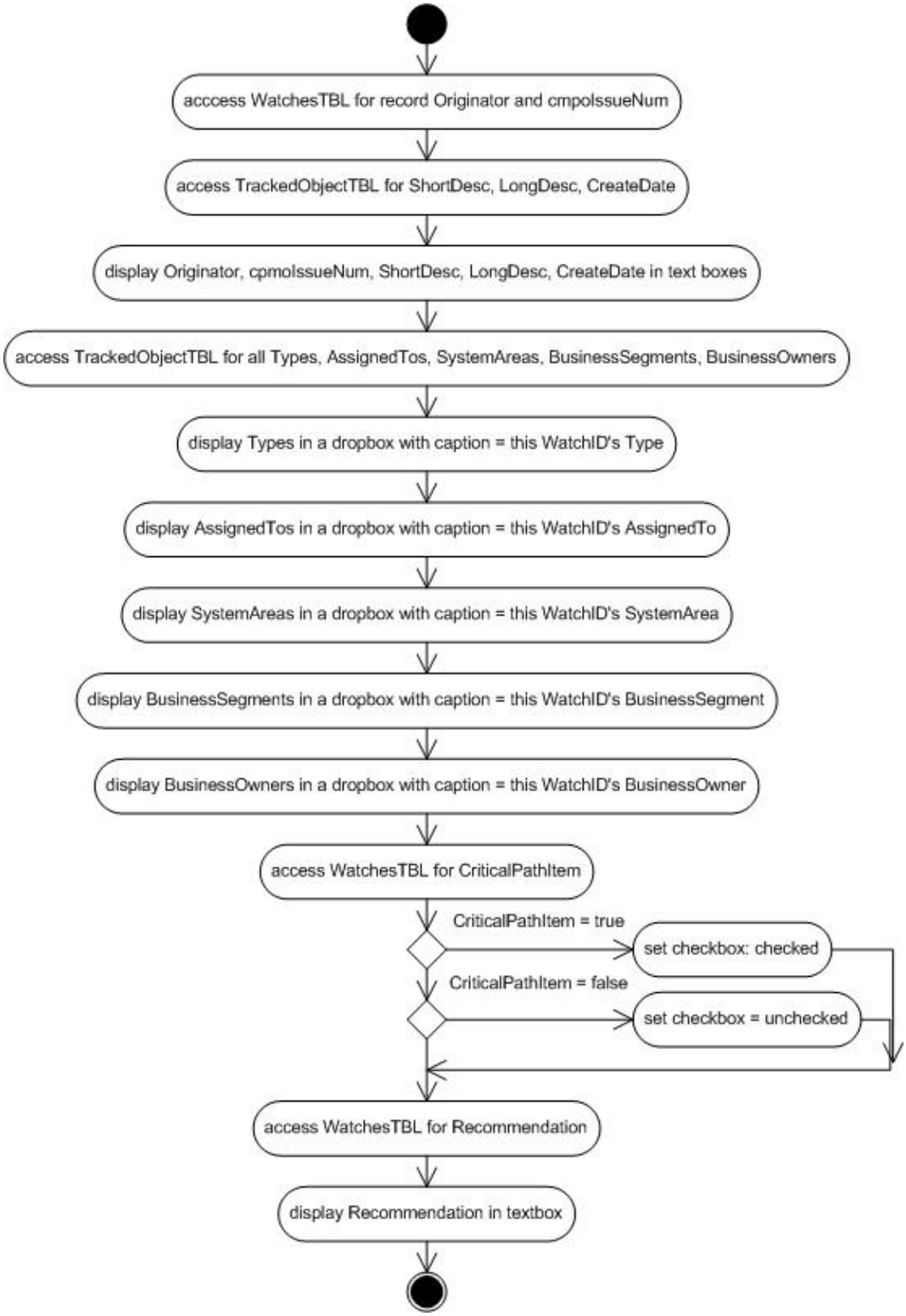
4.27.3 Component Processing Detail

4.27.3.1 Algorithmic Model

```
        In: WatchID
        In: Originator, CreateDate, ShortDesc, LongDesc
        In: Type, AssignedTo
        In: CriticalPathItem = true, false
        In: cpmoIssueNum, Recommendation
        In: SystemArea, BusinessSegment, BusinessOwner
        Out: message of success or message of failure

ModifyWatch( In: WatchID )
        On Error: send message of failure and exit the routine
        lookup from WatchesTBL -> Originator, cmpoIssueNum
        lookup from TrackedObjectTBL -> ShortDesc, LongDesc, CreateDate
        Display Originator, cmpoIssueNum, ShortDesc, LongDesc, CreateDate in text boxes
        For each ID in the TrackedObjectTBL, populate a separate list for each of the following:
                Type (caption = this WatchId's type)
                AssignedTo (caption = this WatchID's AssignedTo)
                SystemArea (caption = this WatchID's SystemArea)
                BusinessSegment (caption = this WatchID's BusinessSegment)
                BusinessOwner (caption = this WatchID's BusinessOwner)
        lookup from WatchesTBL -> CriticalPathItem, Recommendation
        if CriticalPathItem = true
                set the corresponding checkbox control to checked
        else
                set it to unchecked
        populate a text box with Recommendation

Save( In: All from above
      Out: message )
        validate inputs
        On Error: send message of failure and exit the routine
        record to WatchesTBL -> Originator, cmpoIssueNum
        record to TrackedObjectTBL -> ShortDesc, LongDesc, CreateDate
        record to TrackedObjectTBL -> Type, AssignedTo, SystemArea, BusinessSegment,
                  BusinessOwner
        record to WatchesTBL -> CriticalPathItem
        record to WatchesTBL -> recommendation
        if no errors occurred:
```

```
send message of success
```

4.27.3.2 Activity diagram for Modify Watch Module


acccess WatchesTBL for record Originator and cmpoIssueNum
access TrackedObjectTBL for ShortDesc, LongDesc, CreateDate
display Originator, cpmoIssueNum, ShortDesc, LongDesc, CreateDate in text boxes
access TrackedObjectTBL for all Types, AssignedTos, SystemAreas, BusinessSegments, BusinessOwners
display Types in a dropbox with caption = this WatchID's Type
display AssignedTos in a dropbox with caption = this WatchID's AssignedTo
display SystemAreas in a dropbox with caption = this WatchID's SystemArea
display BusinessSegments in a dropbox with caption = this WatchID's BusinessSegment
display BusinessOwners in a dropbox with caption = this WatchID's BusinessOwner
access WatchesTBL for CriticalPathItem
CriticalPathItem = true
set checkbox: checked
CriticalPathItem = false
set checkbox = unchecked
access WatchesTBL for Recommendation
display Recommendation in textbox

4.28 Description of Modify Inventory Object

This component will allow a user to modify an inventory object that is in the inventory list.

4.28.1 Processing Narrative

A user wishes to modify an inventory object in the inventory list. The user must be authenticated and authorized to do so. To get to the inventory window, the user must select the *inventory* tab on the main screen. There will be a command button called *modify inventory*. This will take the user to the modify inventory interface, described below.

4.28.2 Interface Description

On this form, there will be both required and non−required fields. The required elements on this form will be described as follows. There will be text boxes for the following things: the date created, a short description (or title), and a long description. There will be a dropdown list of users (populated from the user table) to select who this object is assigned to. There will also be a dropdown list to select this object's type (detailed design, conceptual design, etc). All of these will be configurable.

The non−required elements on this form will be described as follows. There will be a text boxes to enter the release, the planned draft date, the actual draft date, the planned internal review date, the actual internal review date, the planned business review date, the actual business review date, the planned quality review date, and the actual quality review date. There will be dropdown lists of the name of the team (populated from the team table), the business segment, the system area, and the business owner. There will also be controls to add application systems, and technologies used. All of these things are editable. There will be a *cancel* command button that will enable the user to cancel the process of modifying an inventory item at any time. There will be a *save* button that will save the changes to the database.

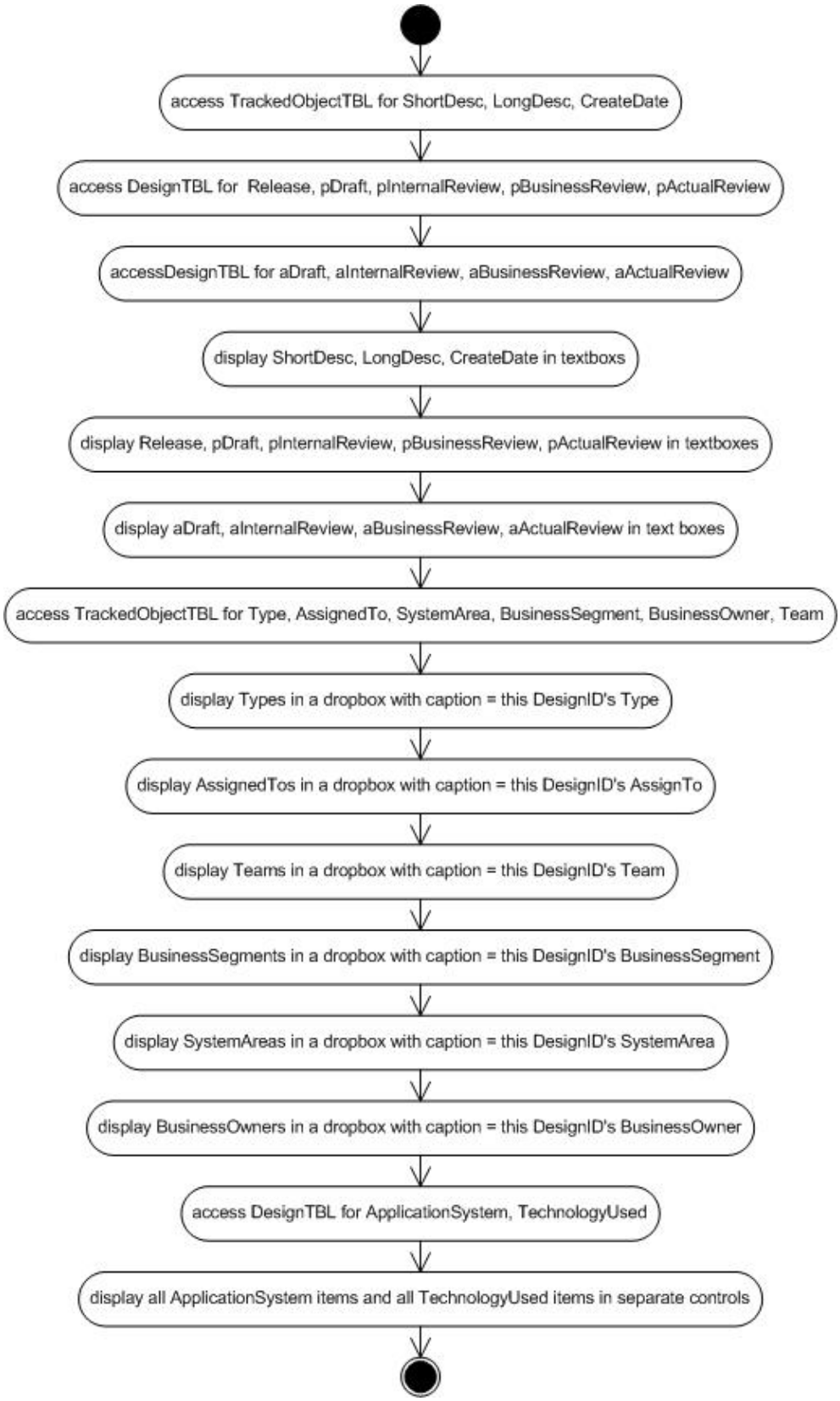
4.28.3 Component Processing Detail

4.28.3.1 Algorithmic Model

```
        In: DesignID
        In: DateCreated, ShortDesc, LongDesc
        In: Type, AssignedTo
        In: Release, PlannedDraftDate, ActualDraftDate, PlannedInternalReviewDate,
            ActualInternalReviewDate, PlannedBusinessReviewDate, ActualBusinessReviewDate,
            PlannedQualityReviewDate, ActualQualityReviewDate
        In: Team, BusinessSegment, SystemArea, BusinessOwner
        In: ApplicationSystem, TechnologyUsed
        Out: message of success or message of failure

ModifyWatch( In: DesignID )
        On Error: send message of failure and exit the routine
        lookup from TrackedObjectTBL -> DateCreated, ShortDesc, LongDesc
        lookup from DesignTBL -> Release, PlannedDraftDate, ActualDraftDate,
                        PlannedInternalReviewDate, ActualInternalReviewDate,
                        PlannedBusinessReviewDate, ActualBusinessReviewDate,
                        PlannedQualityReviewDate, ActualQualityReviewDate
        Display DateCreated, ShortDesc, LongDesc, Release, PlannedDraftDate, ActualDraftDate,
                        PlannedInternalReviewDate, ActualInternalReviewDate, PlannedBusinessRevie
                        ActualBusinessReviewDate, PlannedQualityReviewDate, and ActualQualityRevi
                in text boxes
```

```
        For each ID in the TrackedObjectTBL, populate a separate list for each of the following:
                Type (caption = this DesignID's type)
                AssignedTo (caption = this DesignID's AssignedTo)
                Team (caption = this DesignID's team)
                BusinessSegment (caption = this DesignID's BusinessSegment)
                SystemArea (caption = this DesignID's SystemArea)
                BusinessOwner (caption = this DesignID's BusinessOwner)
        for each ID in the DesignTBL, populate a separate list for each of the following:
                ApplicationSystem
                TechnologyUsed

Save( In: All from above
      Out: message )
        validate inputs
        On Error: send message of failure and exit the routine
        record to TrackedObjectTBL -> DateCreated, ShortDesc, LongDesc
        record to DesignTBL -> Release, PlannedDraftDate, ActualDraftDate,
                    PlannedInternalReviewDate, ActualInternalReviewDate,
                    PlannedBusinessReviewDate, ActualBusinessReviewDate,
                    PlannedQualityReviewDate, ActualQualityReviewDate
        record to TrackedObjectTBL -> Type, AssignedTo, Team, BusinessSegment,
                    SystemArea, BusinessOwner
        record to DesignTBL -> ApplicationSystem, TechnologyUsed
        if no errors occurred:
                send message of success
```

4.28.3.2 Activity diagram for Modify Inventory Module


access TrackedObjectTBL for ShortDesc, LongDesc, CreateDate
access DesignTBL for Release, pDraft, pInternalReview, pBusinessReview, pActualReview
accessDesignTBL for aDraft, aInternalReview, aBusinessReview, aActualReview
display ShortDesc, LongDesc, CreateDate in textboxs
display Release, pDraft, pInternalReview, pBusinessReview, pActualReview in textboxes
display aDraft, aInternalReview, aBusinessReview, aActualReview in text boxes
access TrackedObjectTBL for Type, AssignedTo, SystemArea, BusinessSegment, BusinessOwner, Team
display Types in a dropbox with caption = this DesignID's Type
display AssignedTos in a dropbox with caption = this DesignID's AssignTo
display Teams in a dropbox with caption = this DesignID's Team
display BusinessSegments in a dropbox with caption = this DesignID's BusinessSegment
display SystemAreas in a dropbox with caption = this DesignID's SystemArea
display BusinessOwners in a dropbox with caption = this DesignID's BusinessOwner
access DesignTBL for ApplicationSystem, TechnologyUsed
display all ApplicationSystem items and all TechnologyUsed items in separate controls

4.29 Description of Delete Watch Object

This component will allow a user to delete a watch list object.

4.29.1 Processing Narrative

A user wishes to delete a watch list item. The user must be authenticated and authorized to do so. To get to the watches window, the user must select the *watches* tab on the main screen. There will be a command button called *delete watch*. This will take the user to the delete watch interface, described below.

4.29.2 Interface Description

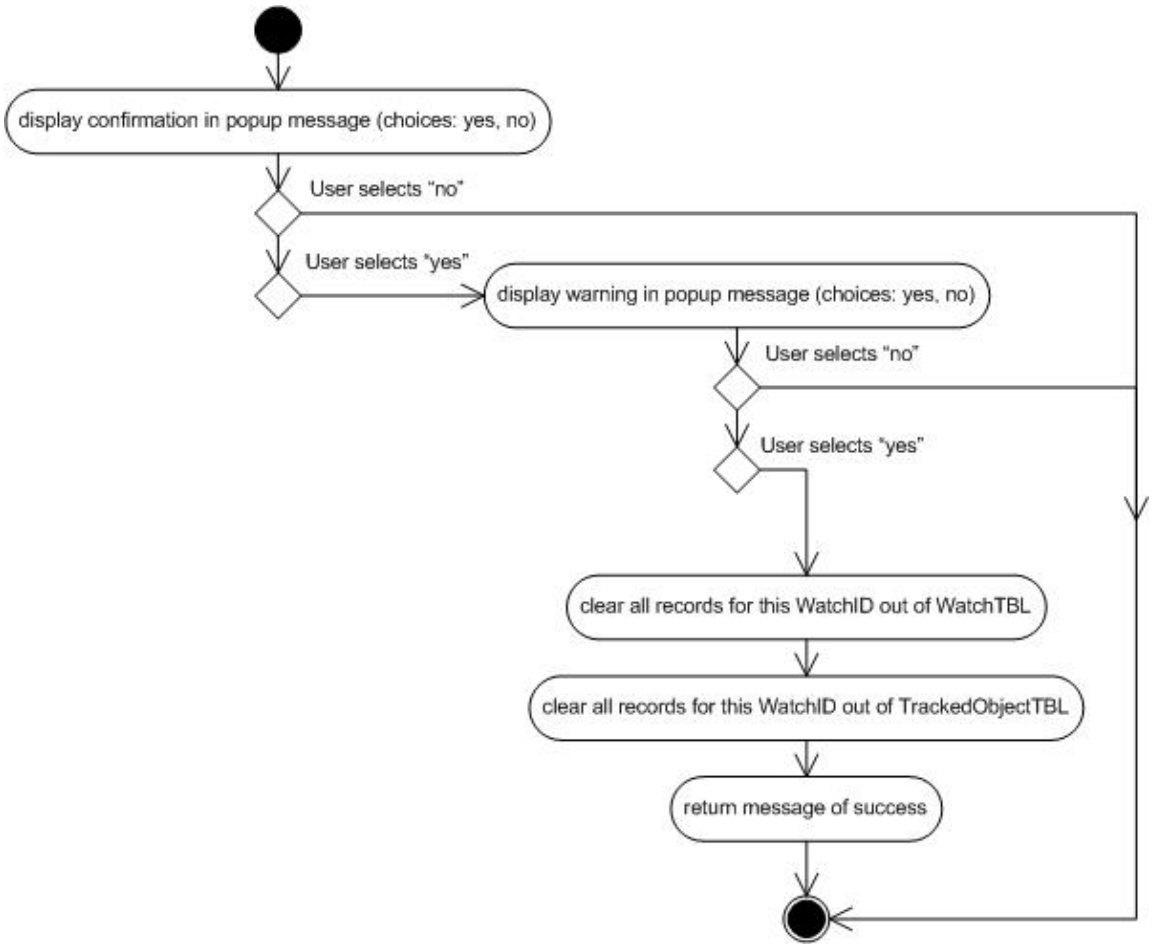
A confirmation message box will appear on the screen. If the user selects *no*, nothing will happen. If the user selects *Yes*, another message box will appear on the screen to remind the user to make sure nothing depends on this object, and to ask them if they are sure they wish to delete the object. If the user selects *no*, nothing will happen. If the user selects *Yes*, the object will be deleted.

4.29.3 Component Processing Detail

4.29.3.1 Algorithmic Model

```
        In: WatchID
        In: Confirm = yes, no
        In: Warning = yes, no
        Out: message of success or message of failure

Yes( Out: message )
        On Error: send message of failure and exit the routine
        clear all records from WatchTBL where WatchID is this WatchID
        clear all records from TrackedObjectTBL where ID is this WatchID
        if no errors occurred:
                send message of success

No( )
        do nothing
```

4.29.3.2 Activity diagram for Delete Watch Module


display confirmation in popup message (choices: yes, no)
User selects "no"
User selects "yes"
display warning in popup message (choices: yes, no)
User selects "no"
User selects "yes"
clear all records for this WatchID out of WatchTBL
clear all records for this WatchID out of TrackedObjectTBL
return message of success

4.30 Description of Delete Inventory Object

This component will allow a user to delete an inventory object.

4.30.1 Processing Narrative

A user wishes to delete an inventory object. The user must be authenticated and authorized to do so. To get to the inventory window, the user must select the *inventory* tab on the main screen. There will be a command button called *delete inventory*. This will take the user to the delete inventory interface, described below.

4.30.2 Interface Description

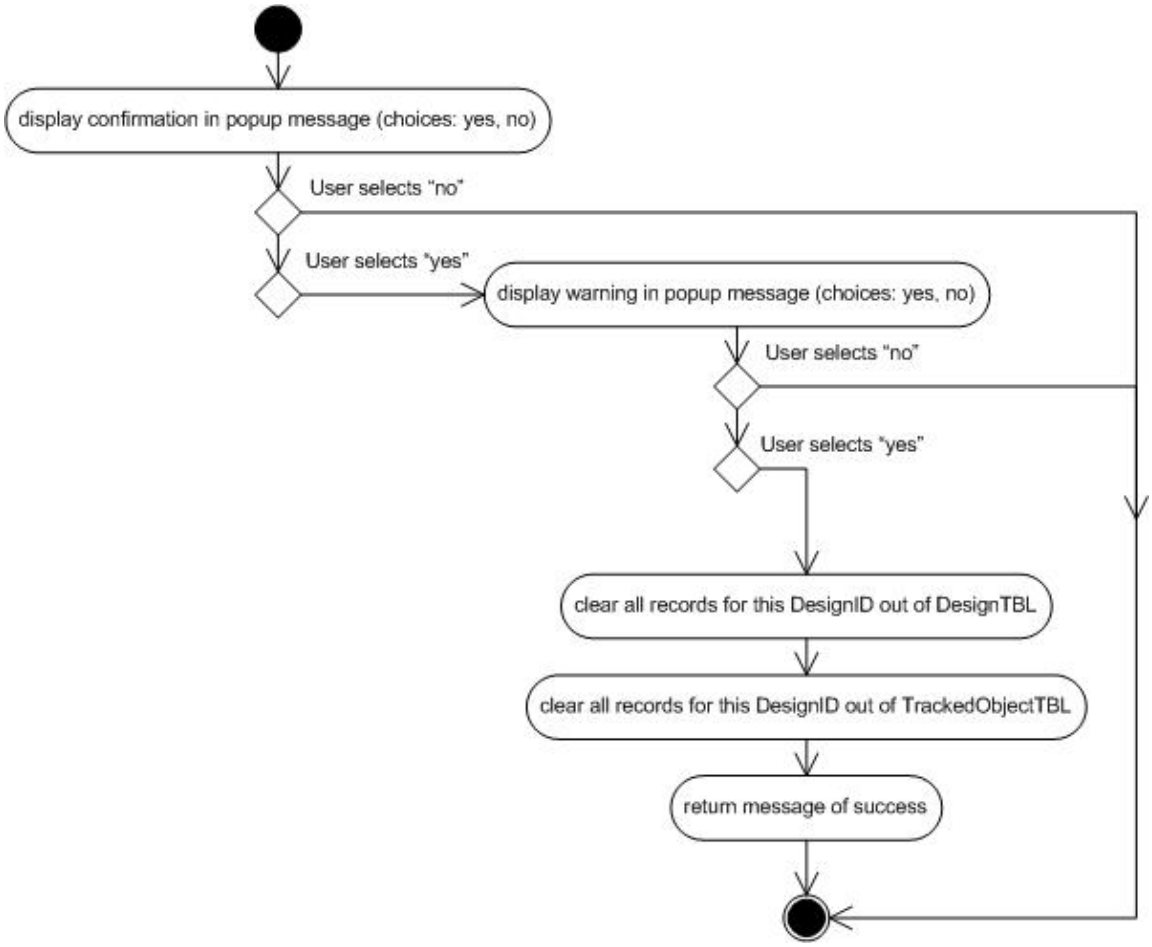
A confirmation message box will appear on the screen. If the user selects *no*, nothing will happen. If the user selects *Yes*, another message box will appear on the screen to remind the user to make sure nothing depends on this object, and to ask them if they are sure they wish to delete the object. If the user selects *no*, nothing will happen. If the user selects *Yes*, the object will be deleted.

4.30.3 Component Processing Detail

4.30.3.1 Algorithmic Model

```
        In: DesignID
        In: Confirm = yes, no
        In: Warning = yes, no
        Out: message of success or message of failure

Yes( Out: message )
        On Error: send message of failure and exit the routine
        clear all records from DesignTBL where ID is this DesignID
        clear all records from TrackedObjectTBL where ID is this DesignID
        if no errors occurred:
                send message of success

No( )
        do nothing
```

4.30.3.2 Activity diagram for Delete Inventory Module


display confirmation in popup message (choices: yes, no)
User selects "no"
User selects "yes"
display warning in popup message (choices: yes, no)
User selects "no"
User selects "yes"
clear all records for this DesignID out of DesignTBL
clear all records for this DesignID out of TrackedObjectTBL
return message of success

4.31 Description of Customized Report

This component will allow a user to make and view a customized report.

4.31.1 Processing Narrative

A user wishes to make a customized report. The user must be authenticated and authorized to do so. To get to the reports window, the user must select the *Reports* tab on the main screen.

4.31.2 Interface Description

On this form, there will be a section for Preconfigured Reports, and a section for Customized Reports. The user will go to the Customized Reports section. The user will choose from Watch List, Inventory, and Requirements. They will then have a view of all of the attributes of their selection. They may choose any of these attributes and select the *View* button. A report will be generated based on their selections. There will also be a button to save this report. Once a report is saved, it will appear on the list of Preconfigured Reports.

4.31.3 Component Processing Detail

4.31.3.1 Algorithmic Model

```
        In: Selection1 = WatchList, Inventory, Requirements
        In: Selection2 -- SelectionN = attributes of Selection1
        Out: A report based on Selection1 and Selection2 - SelectionN

FormLoad( )
        On Error: send message of failure and exit the routine
        if Selection1 = WatchList
                lookup WatchList attributes
                display WatchList attributes in lists
        if Selection1 = Inventory
                lookup Inventory  attributes
                display Inventory attributes in lists
        if Selection1 = Requirements
                lookup Requirements attributes
                display Requirements attributes in lists

        if no errors occurred:
                send message of success

View( In: all from above
      Out: a report )
        Either a report is shown with the selected fields filled in, or the form designer will  a

Save( )
        The user will enter a name for the report
        The user will enter a short description for the report
        The report information will be saved
```

4.32 Description of View Preconfigured Report

This component will allow a user to view a preconfigured report.

4.32.1 Processing Narrative

A user wishes to view a preconfigured report. The user must be authenticated and authorized to do so. To get

to the reports window, the user must select the *Reports* tab on the main screen.

4.32.2 Interface Description

On this form, there will be a section for Preconfigured Reports, and a section for Customized Reports. The user will go to the Preconfigured Reports section. The user will see a list of reports. They will select a report from this list and select the *view* button.

4.32.3 Component Processing Detail

4.32.3.1 Algorithmic Model

```
        In: Selection = ReportName
        Out: A report based on Selection

View( In: ReportName
           Out: a report )
        The report information is loaded from the database, and a report will be displayed
```

5 Sequence Diagrams

The Use Cases for the tracking tool are:

1. Add Object
2. Authenticate User
3. Change Log
4. Delete Object
5. Email to Assigned To Person
6. Generate Reports

    A. Ad−Hoc Reports
    B. Pre−Configured Reports

7.Linking and Embedding

    A. Link an Internal Object
    B. Link an External Object
    C. Attach an External Object

8. Modify Object
9. View Object List
10. View Object

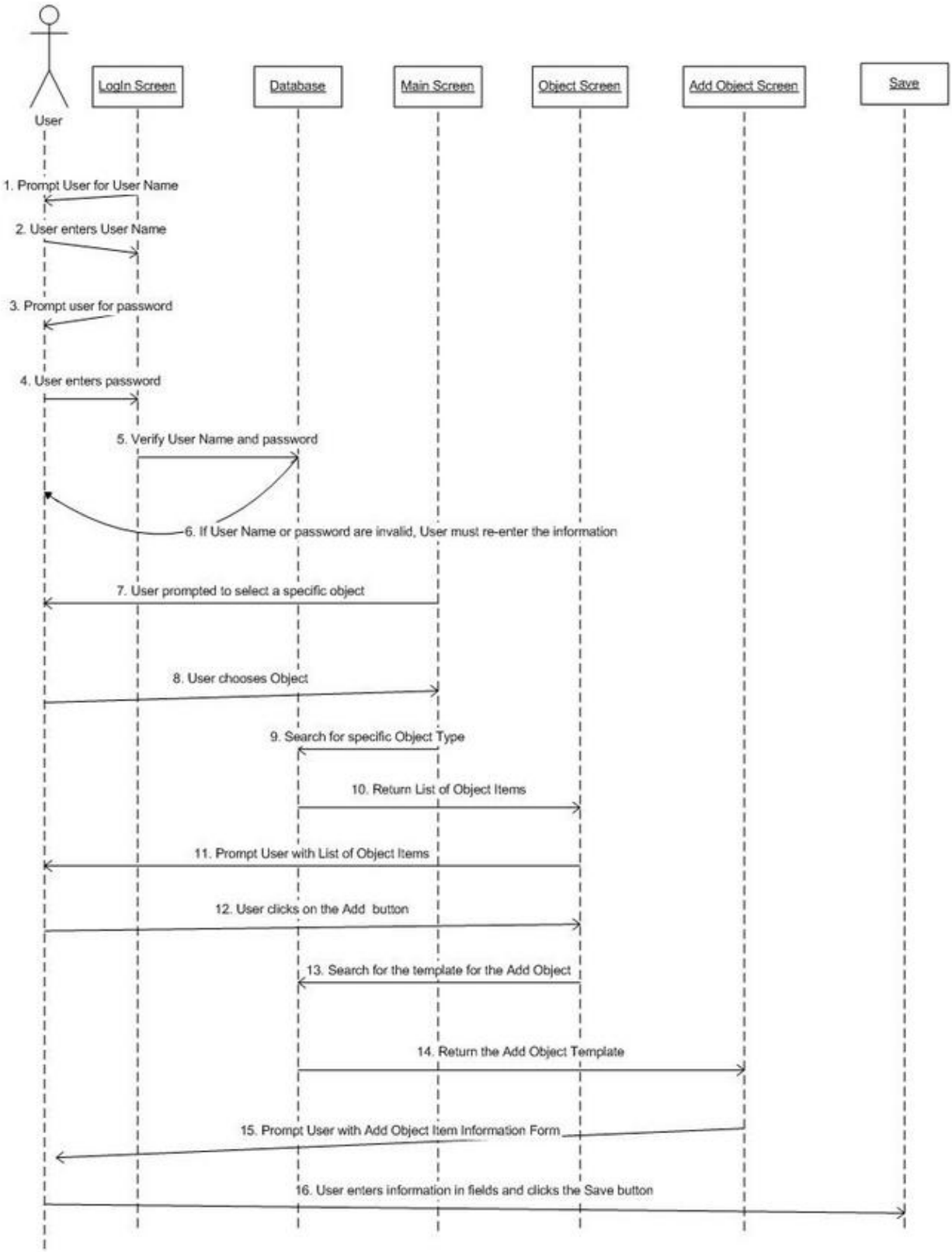
In the use cases, we referred to objects. Objects can be defined as one of the items listed below:

1. Watch Item
2. Inventory Item
3. Users
4. Groups

The following are sequence diagrams for all the use cases listed above.

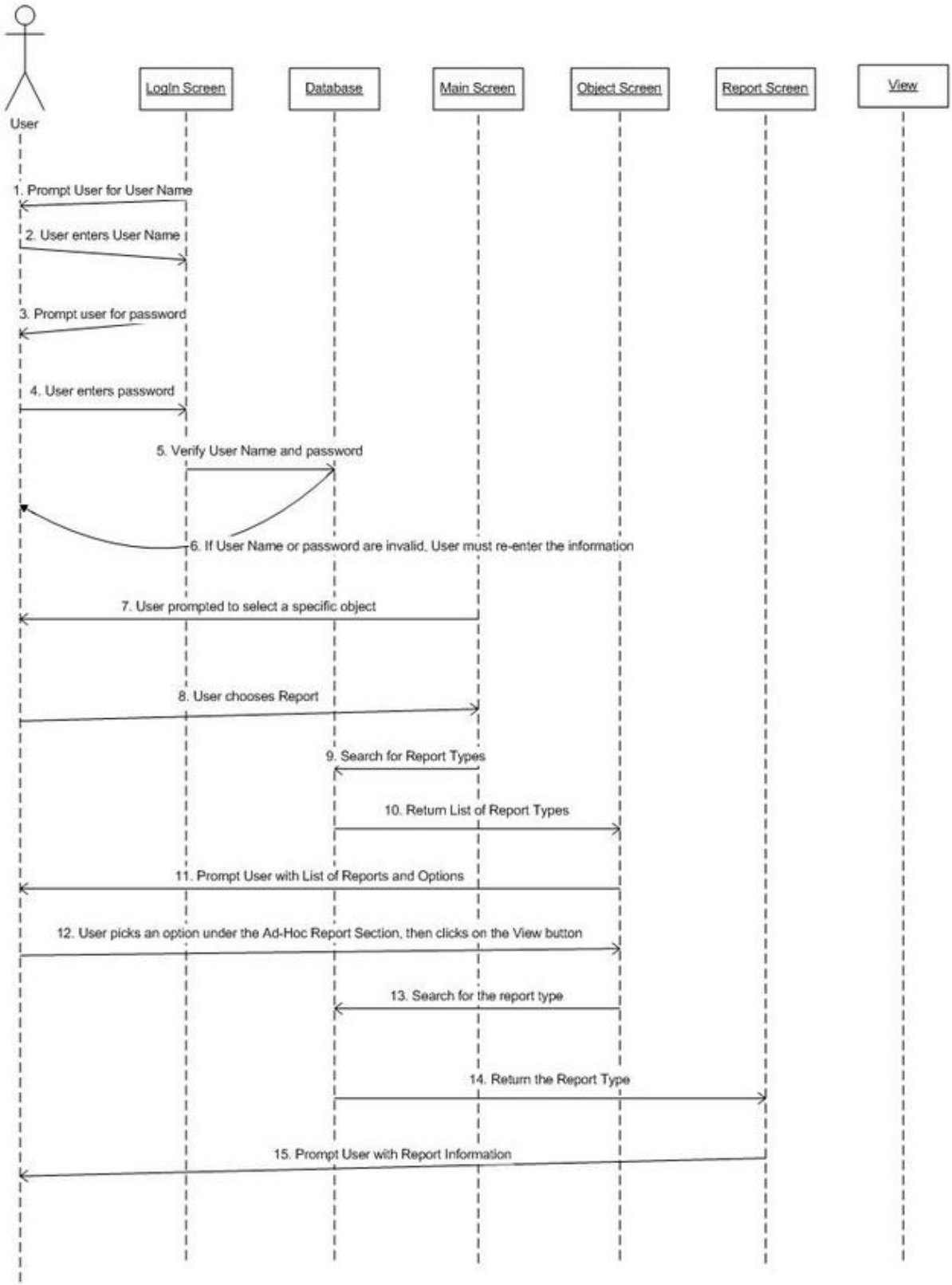
Add Object Sequence Diagram

The user wants to add an object to the tracking tool database. The interaction between the user, tracking tool interface, and the database are described in the following diagram.


User
LogIn Screen
Database
Main Screen
Object Screen
Add Object Screen
Save
1. Prompt User for User Name
2. User enters User Name
3. Prompt user for password
4. User enters password
5. Verify User Name and password
6. If User Name or password are invalid, User must re-enter the information
7. User prompted to select a specific object
8. User chooses Object
9. Search for specific Object Type
10. Return List of Object Items
11. Prompt User with List of Object Items
12. User clicks on the Add button
13. Search for the template for the Add Object
14. Return the Add Object Template
15. Prompt User with Add Object Item Information Form
16. User enters information in fields and clicks the Save button

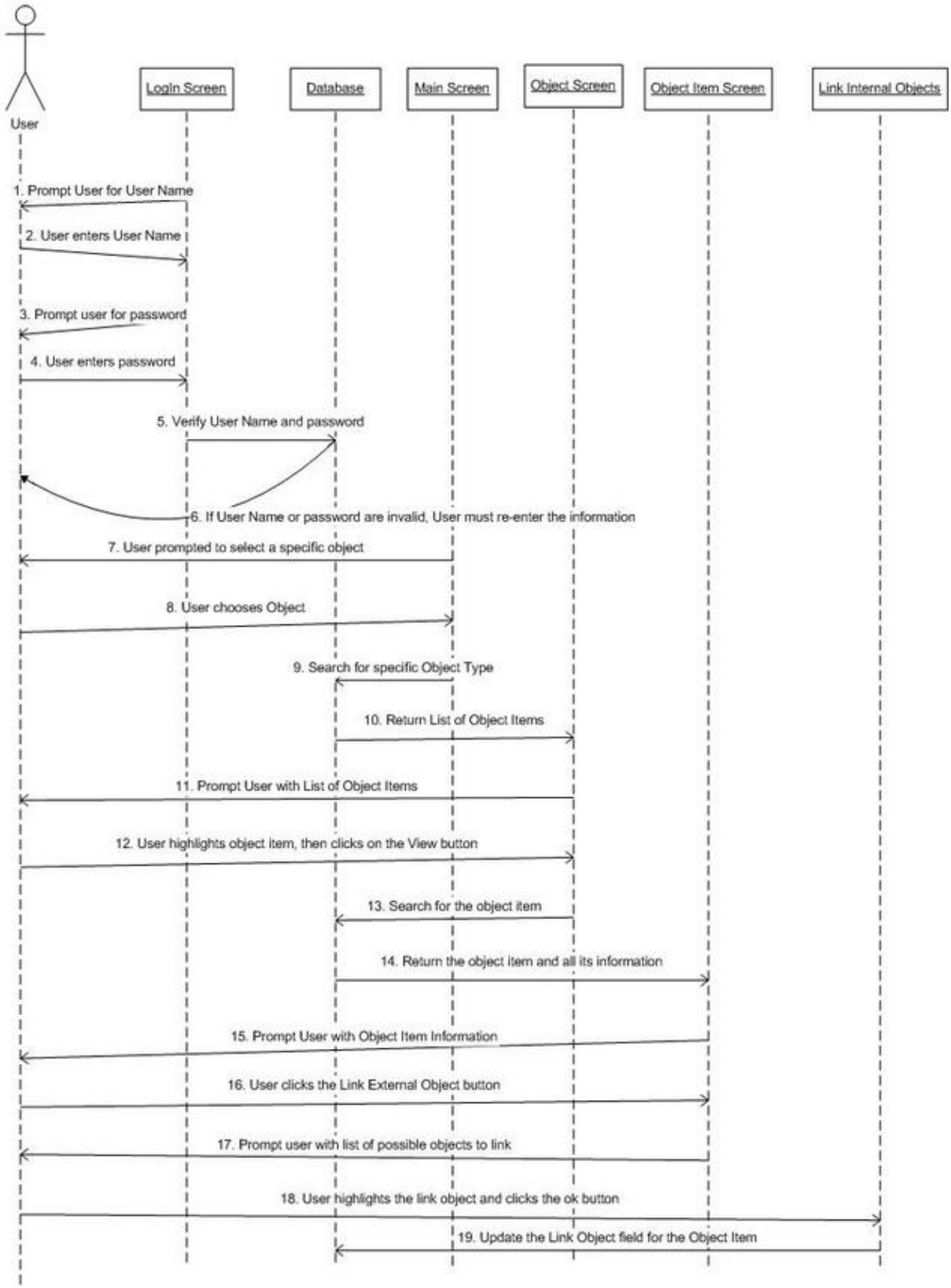
Ad−Hoc Sequence Diagram

The user wants to create a new report using data stored in the tracking tool database. The interaction between the user, tracking tool interface, and the database are described in the following diagram.


User
LogIn Screen
Database
Main Screen
Object Screen
Report Screen
View
1. Prompt User for User Name
2. User enters User Name
3. Prompt user for password
4. User enters password
5. Verify User Name and password
6. If User Name or password are invalid, User must re-enter the information
7. User prompted to select a specific object
8. User chooses Report
9. Search for Report Types
10. Return List of Report Types
11. Prompt User with List of Reports and Options
12. User picks an option under the Ad-Hoc Report Section, then clicks on the View button
13. Search for the report type
14. Return the Report Type
15. Prompt User with Report Information

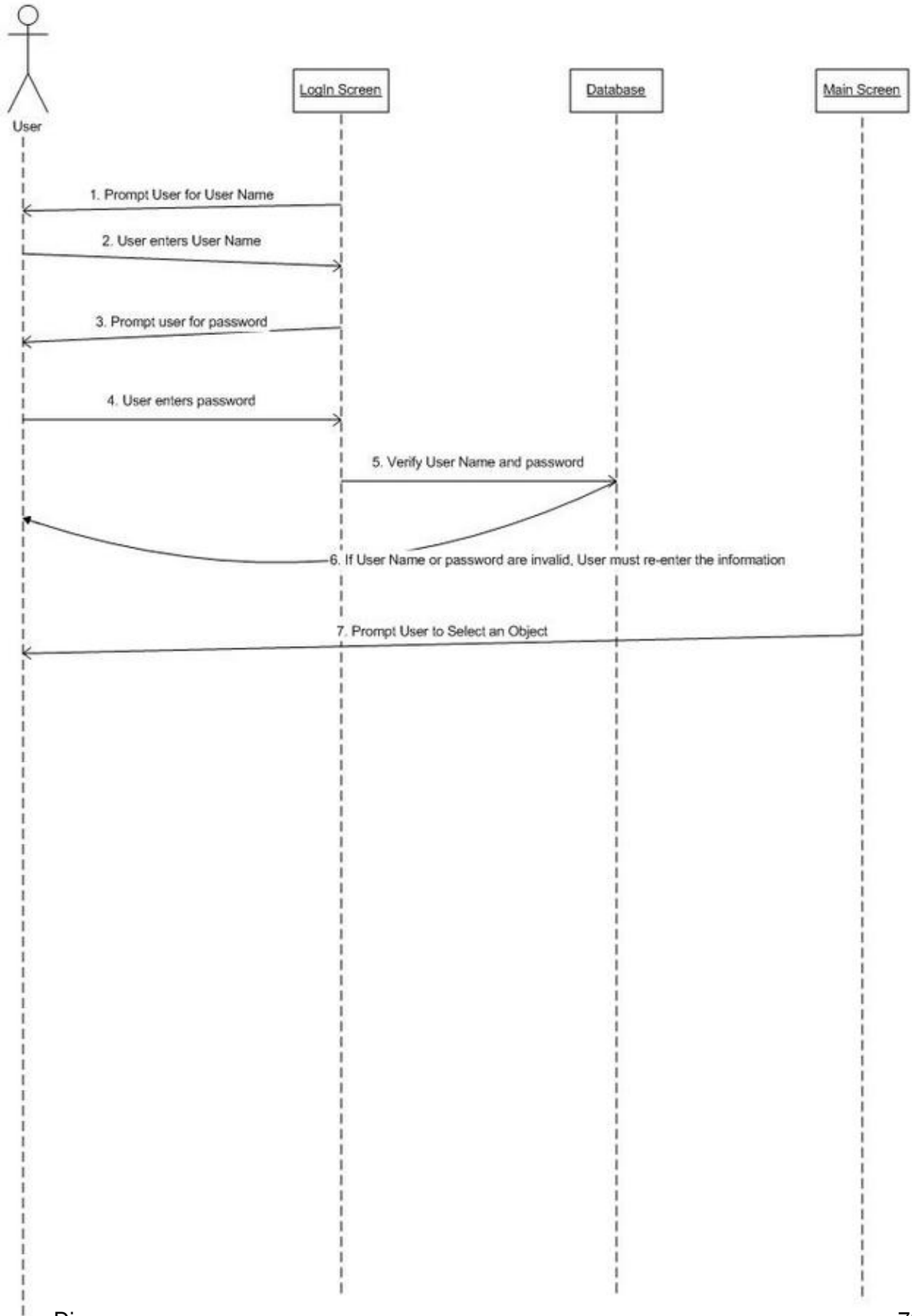
Attach External Object Sequence Diagram

The user wants to attach an external object to an item in the tracking tool database. The interaction between the user, tracking tool interface, and the database are described in the following diagram.


User
LogIn Screen
Database
Main Screen
Object Screen
Object Item Screen
Link Internal Objects
1. Prompt User for User Name
2. User enters User Name
3. Prompt user for password
4. User enters password
5. Verify User Name and password
6. If User Name or password are invalid, User must re-enter the information
7. User prompted to select a specific object
8. User chooses Object
9. Search for specific Object Type
10. Return List of Object Items
11. Prompt User with List of Object Items
12. User highlights object item, then clicks on the View button
13. Search for the object item
14. Return the object item and all its information
15. Prompt User with Object Item Information
16. User clicks the Link External Object button
17. Prompt user with list of possible objects to link
18. User highlights the link object and clicks the ok button
19. Update the Link Object field for the Object Item

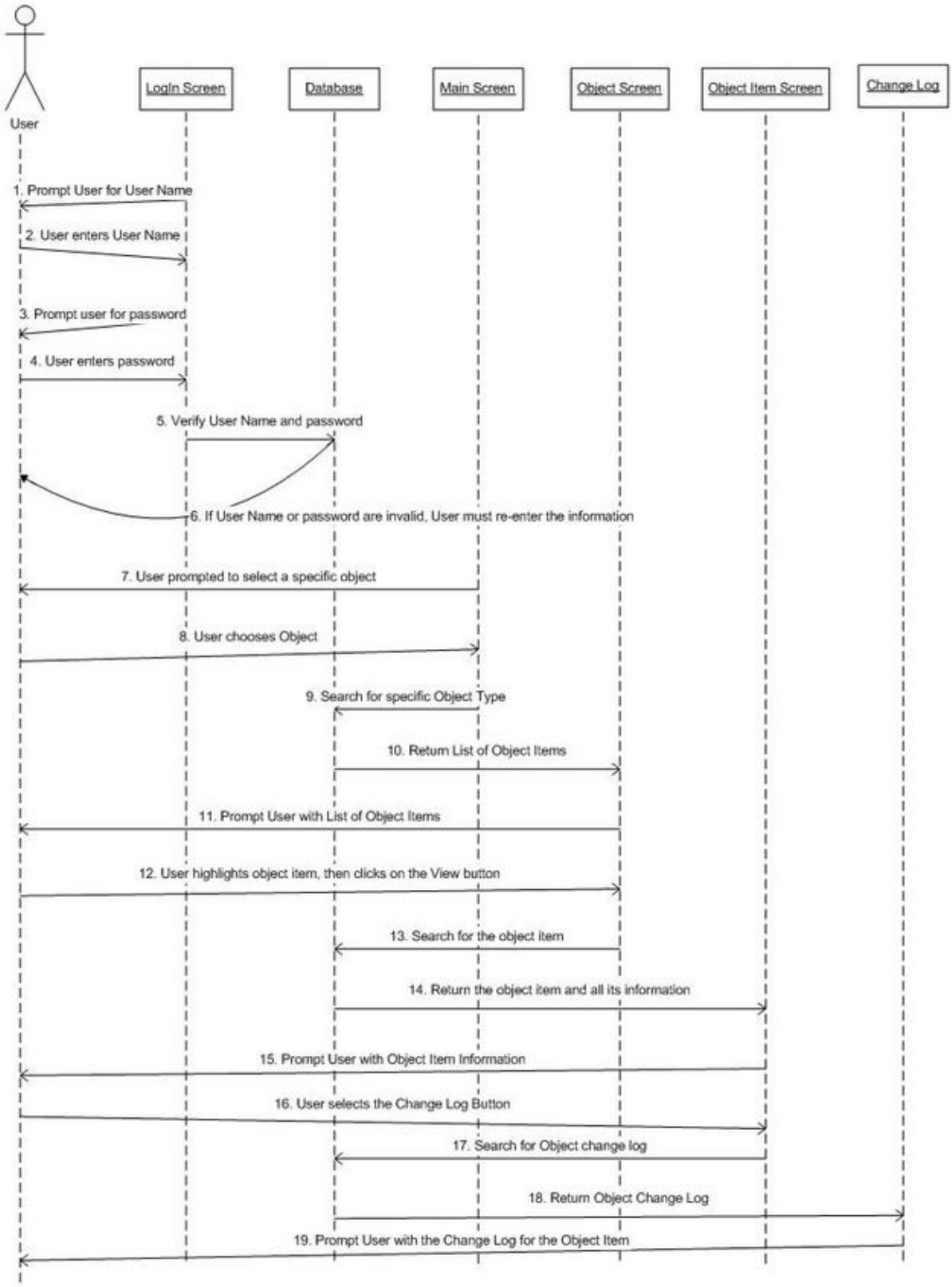
Authenticate User Sequence Diagram

The user wants log into the tracking tool database. The interaction between the user, tracking tool interface, and the database are described in the following diagram.


User
LogIn Screen
Database
Main Screen
1. Prompt User for User Name
2. User enters User Name
3. Prompt user for password
4. User enters password
5. Verify User Name and password
6. If User Name or password are invalid, User must re-enter the information
7. Prompt User to Select an Object

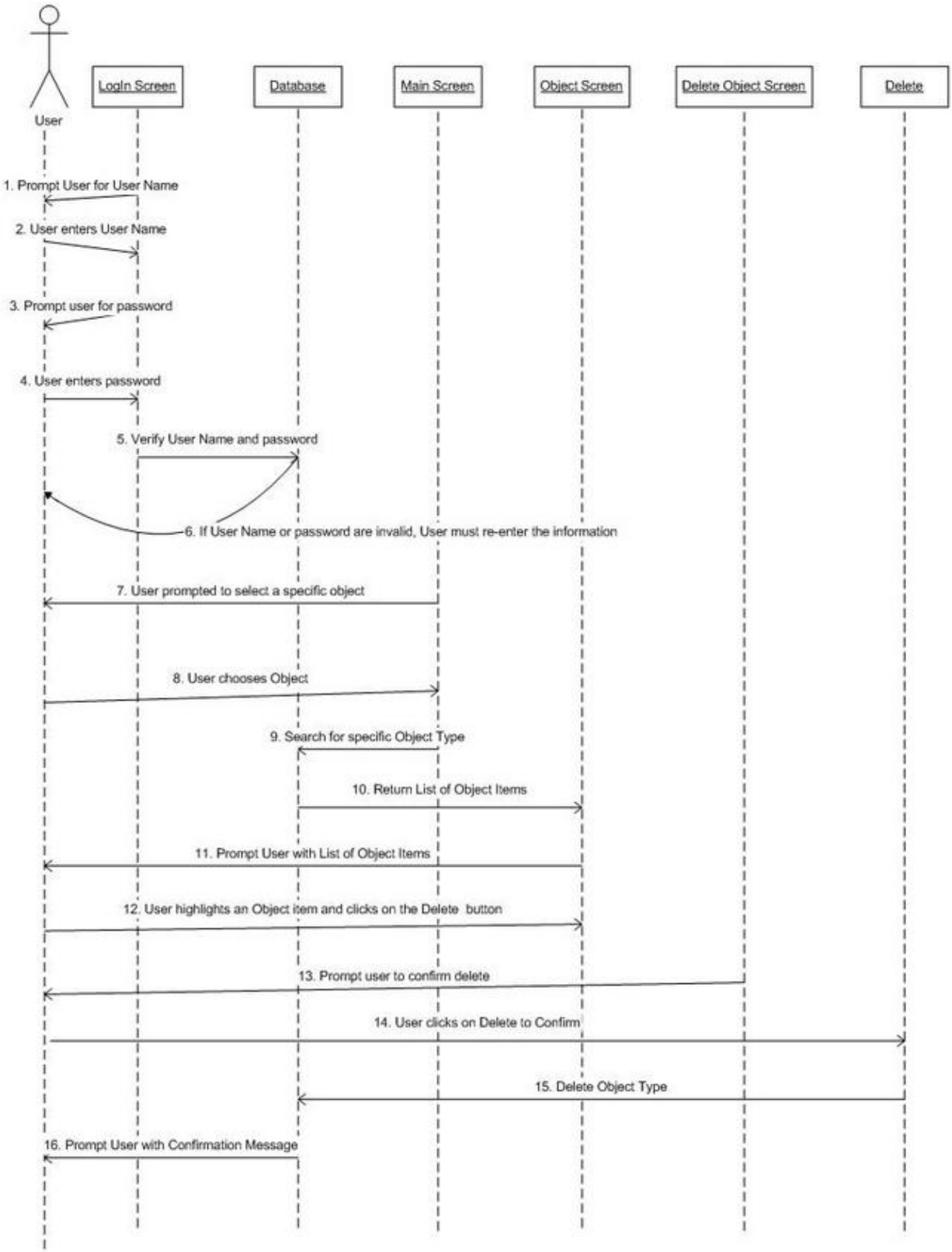
Change Log Sequence Diagram

The user wants to view all the changes made to an item in the tracking tool database. The interaction between the user, tracking tool interface, and the database are described in the following diagram.


User
LogIn Screen
Database
Main Screen
Object Screen
Object Item Screen
Change Log
1. Prompt User for User Name
2. User enters User Name
3. Prompt user for password
4. User enters password
5. Verify User Name and password
6. If User Name or password are invalid, User must re-enter the information
7. User prompted to select a specific object
8. User chooses Object
9. Search for specific Object Type
10. Return List of Object Items
11. Prompt User with List of Object Items
12. User highlights object item, then clicks on the View button
13. Search for the object item
14. Return the object item and all its information
15. Prompt User with Object Item Information
16. User selects the Change Log Button
17. Search for Object change log
18. Return Object Change Log
19. Prompt User with the Change Log for the Object Item

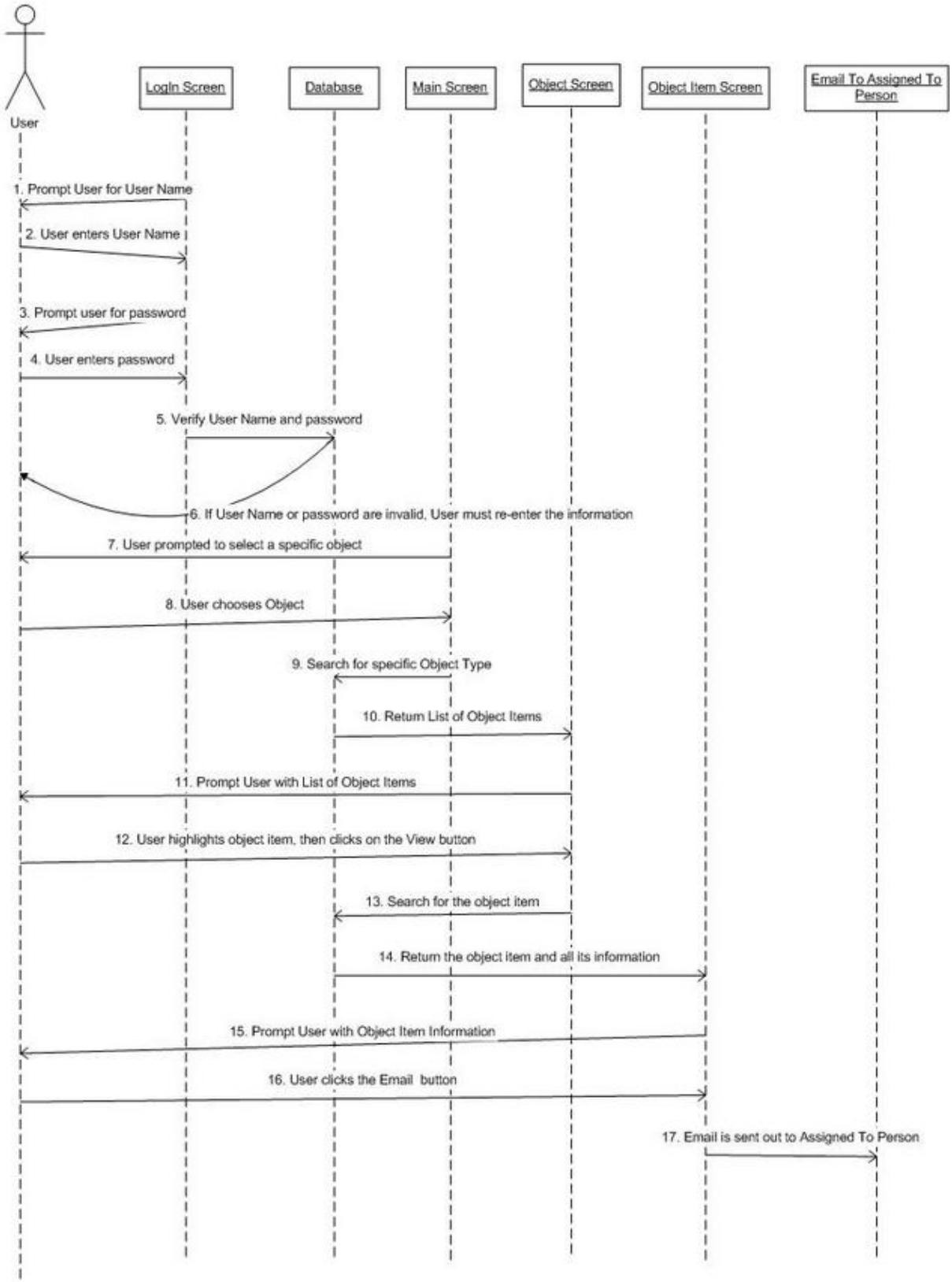
Delete Object Item Sequence Diagram

The user wants to delete an object in the tracking tool database. The interaction between the user, tracking tool interface, and the database are described in the following diagram.


User
LogIn Screen
Database
Main Screen
Object Screen
Delete Object Screen
Delete
1. Prompt User for User Name
2. User enters User Name
3. Prompt user for password
4. User enters password
5. Verify User Name and password
6. If User Name or password are invalid, User must re-enter the information
7. User prompted to select a specific object
8. User chooses Object
9. Search for specific Object Type
10. Return List of Object Items
11. Prompt User with List of Object Items
12. User highlights an Object item and clicks on the Delete button
13. Prompt user to confirm delete
14. User clicks on Delete to Confirm
15. Delete Object Type
16. Prompt User with Confirmation Message

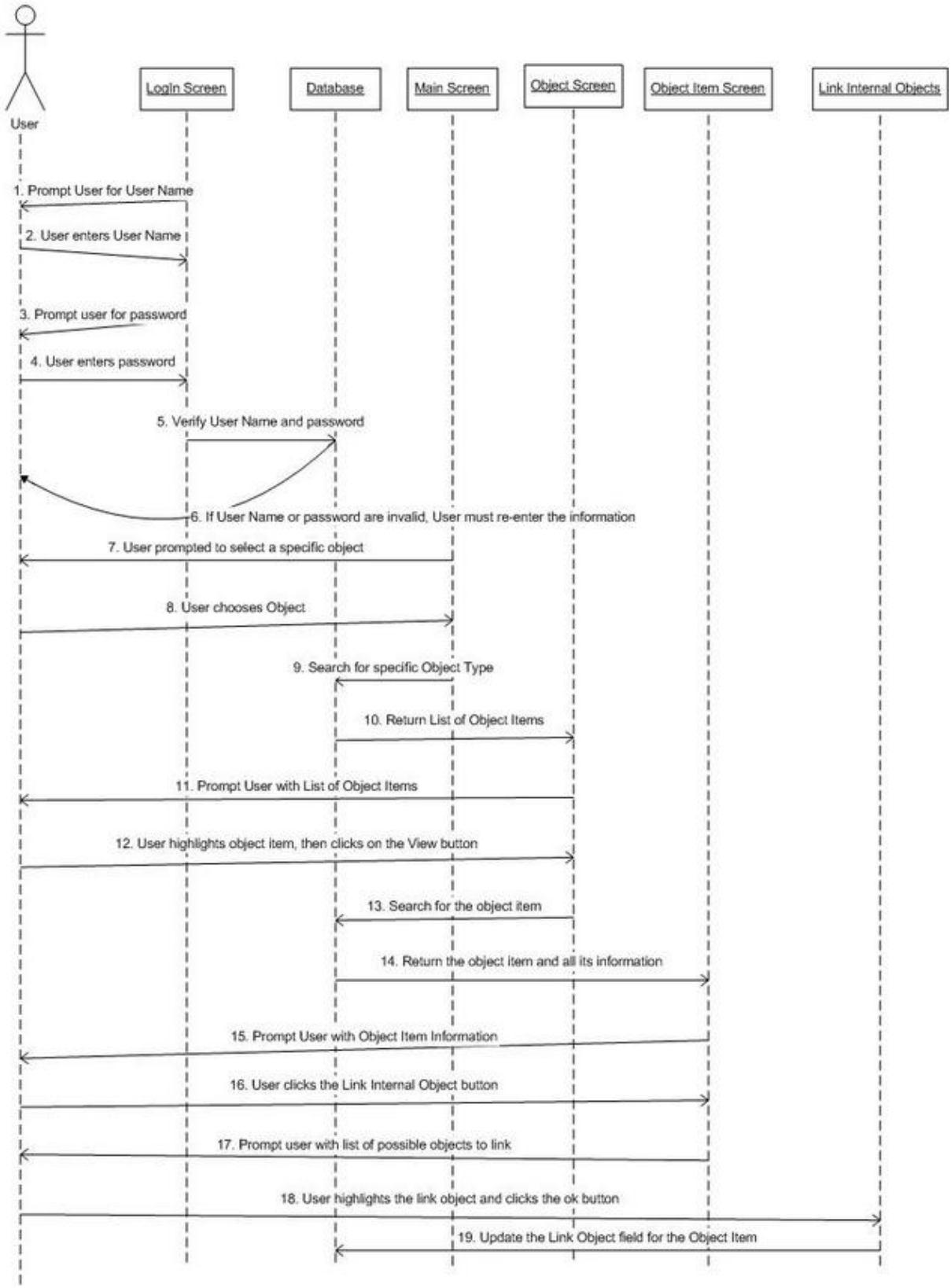
Email to Assigned To Person Sequence Diagram

The user wants to send an email to the person in charge of a specific item in the tracking tool database. The interaction between the user, tracking tool interface, and the database are described in the following diagram.


User
LogIn Screen
Database
Main Screen
Object Screen
Object Item Screen
Email To Assigned To Person
1. Prompt User for User Name
2. User enters User Name
3. Prompt user for password
4. User enters password
5. Verify User Name and password
6. If User Name or password are invalid, User must re-enter the information
7. User prompted to select a specific object
8. User chooses Object
9. Search for specific Object Type
10. Return List of Object Items
11. Prompt User with List of Object Items
12. User highlights object item, then clicks on the View button
13. Search for the object item
14. Return the object item and all its information
15. Prompt User with Object Item Information
16. User clicks the Email button
17. Email is sent out to Assigned To Person

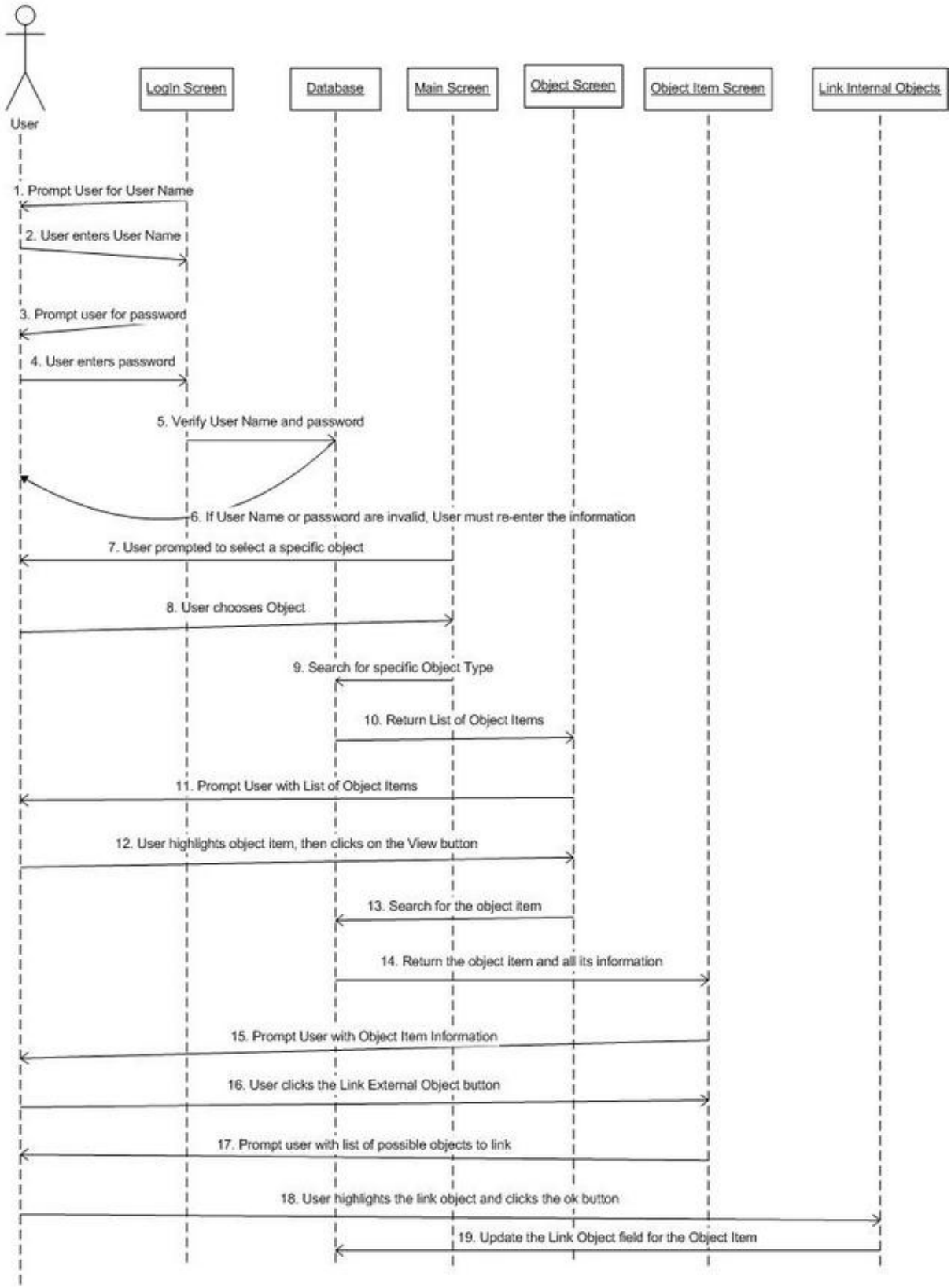
Link Internal Object Sequence Diagram

The user wants to link an internal object to an item in the tracking tool database. The interaction between the user, tracking tool interface, and the database are described in the following diagram.


User
LogIn Screen
Database
Main Screen
Object Screen
Object Item Screen
Link Internal Objects
1. Prompt User for User Name
2. User enters User Name
3. Prompt user for password
4. User enters password
5. Verify User Name and password
6. If User Name or password are invalid, User must re-enter the information
7. User prompted to select a specific object
8. User chooses Object
9. Search for specific Object Type
10. Return List of Object Items
11. Prompt User with List of Object Items
12. User highlights object item, then clicks on the View button
13. Search for the object item
14. Return the object item and all its information
15. Prompt User with Object Item Information
16. User clicks the Link Internal Object button
17. Prompt user with list of possible objects to link
18. User highlights the link object and clicks the ok button
19. Update the Link Object field for the Object Item

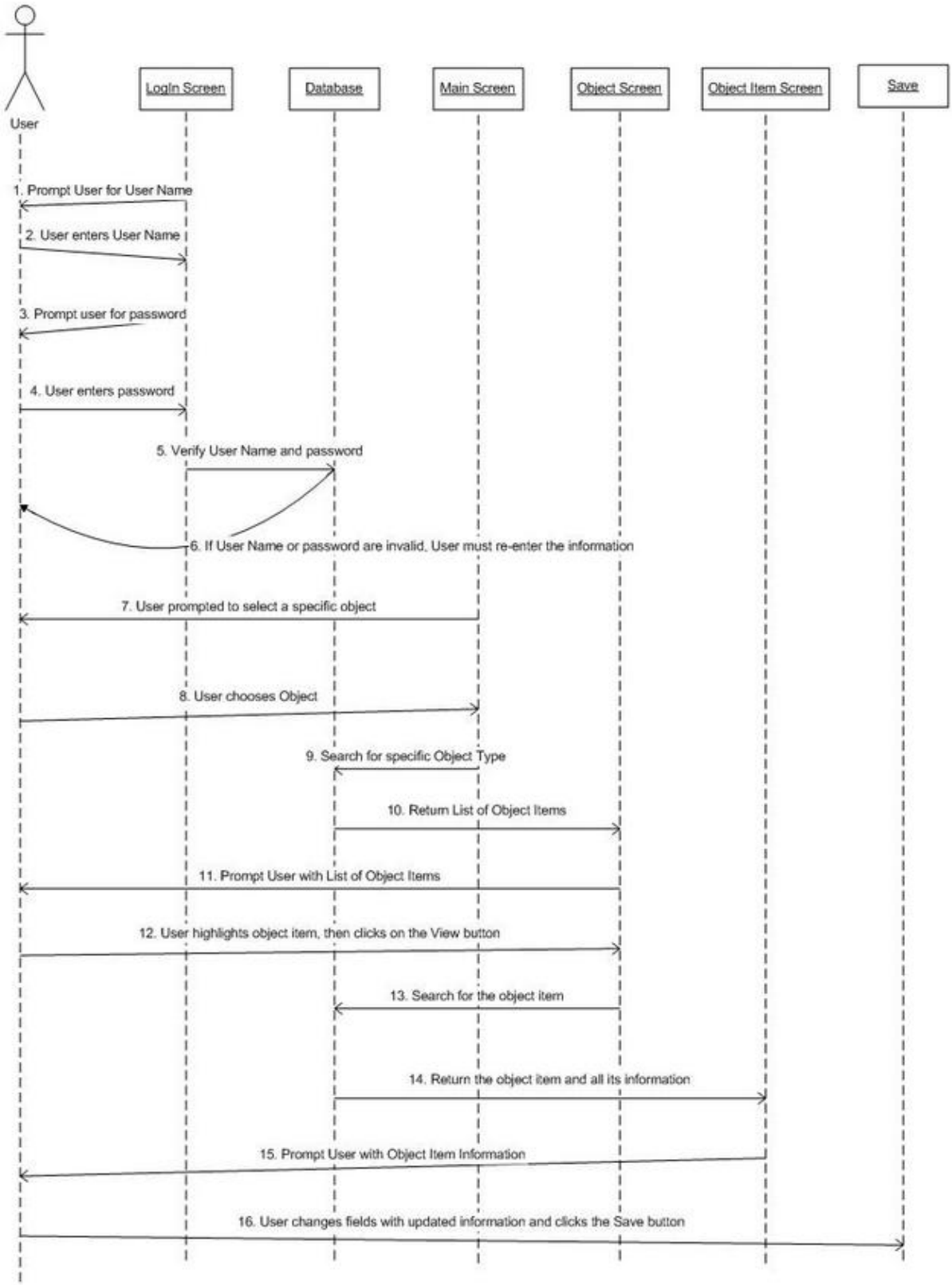
Link External Object Sequence Diagram

The user wants to link an external object to an item in the tracking tool database. The interaction between the user, tracking tool interface, and the database are described in the following diagram.


User
LogIn Screen
Database
Main Screen
Object Screen
Object Item Screen
Link Internal Objects
1. Prompt User for User Name
2. User enters User Name
3. Prompt user for password
4. User enters password
5. Verify User Name and password
6. If User Name or password are invalid, User must re-enter the information
7. User prompted to select a specific object
8. User chooses Object
9. Search for specific Object Type
10. Return List of Object Items
11. Prompt User with List of Object Items
12. User highlights object item, then clicks on the View button
13. Search for the object item
14. Return the object item and all its information
15. Prompt User with Object Item Information
16. User clicks the Link External Object button
17. Prompt user with list of possible objects to link
18. User highlights the link object and clicks the ok button
19. Update the Link Object field for the Object Item

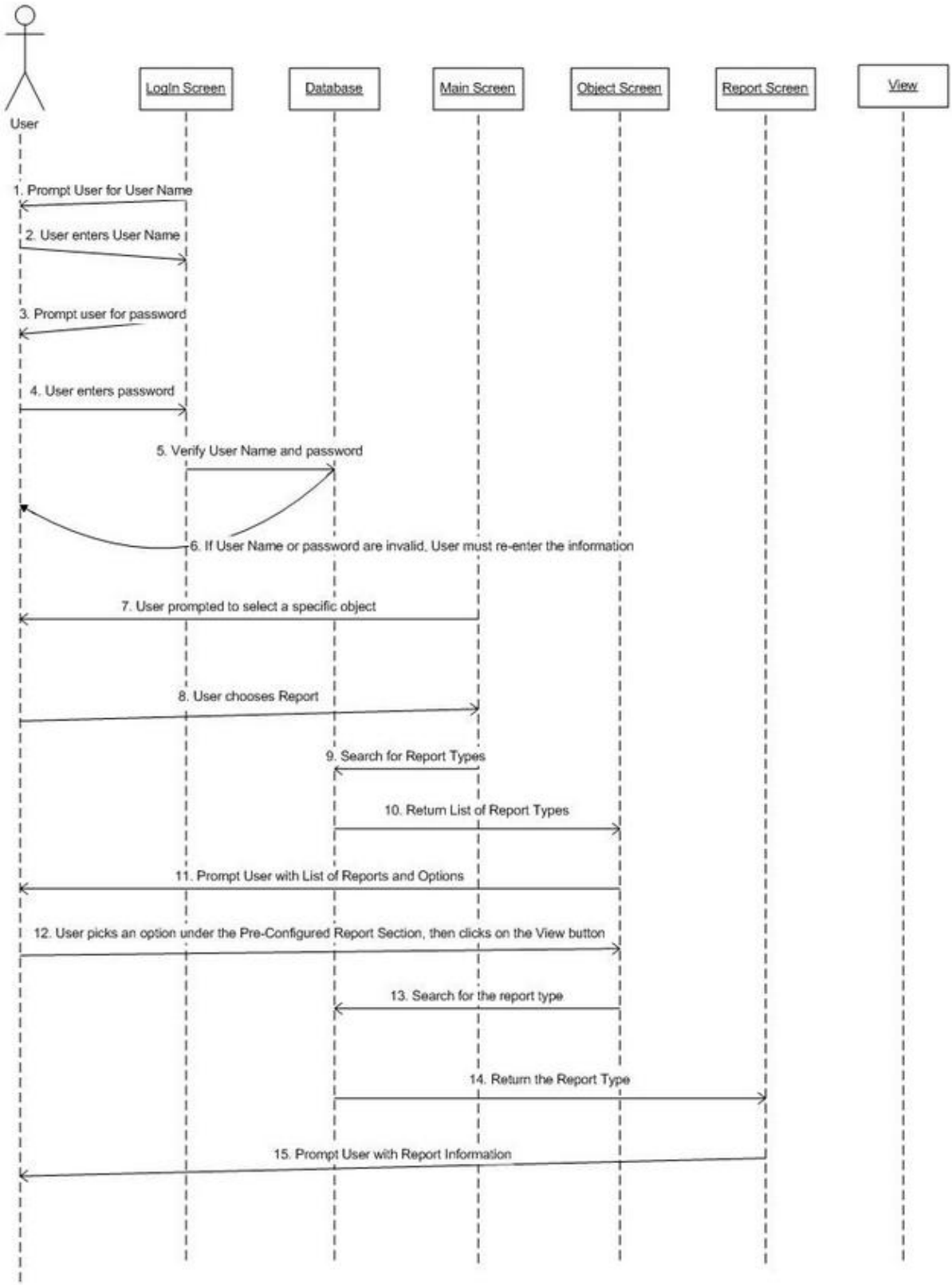
Modify Object Item Sequence Diagram

The user wants to modify an object in the tracking tool database. The interaction between the user, tracking tool interface, and the database are described in the following diagram.


User
LogIn Screen
Database
Main Screen
Object Screen
Object Item Screen
Save
1. Prompt User for User Name
2. User enters User Name
3. Prompt user for password
4. User enters password
5. Verify User Name and password
6. If User Name or password are invalid, User must re-enter the information
7. User prompted to select a specific object
8. User chooses Object
9. Search for specific Object Type
10. Return List of Object Items
11. Prompt User with List of Object Items
12. User highlights object item, then clicks on the View button
13. Search for the object item
14. Return the object item and all its information
15. Prompt User with Object Item Information
16. User changes fields with updated information and clicks the Save button

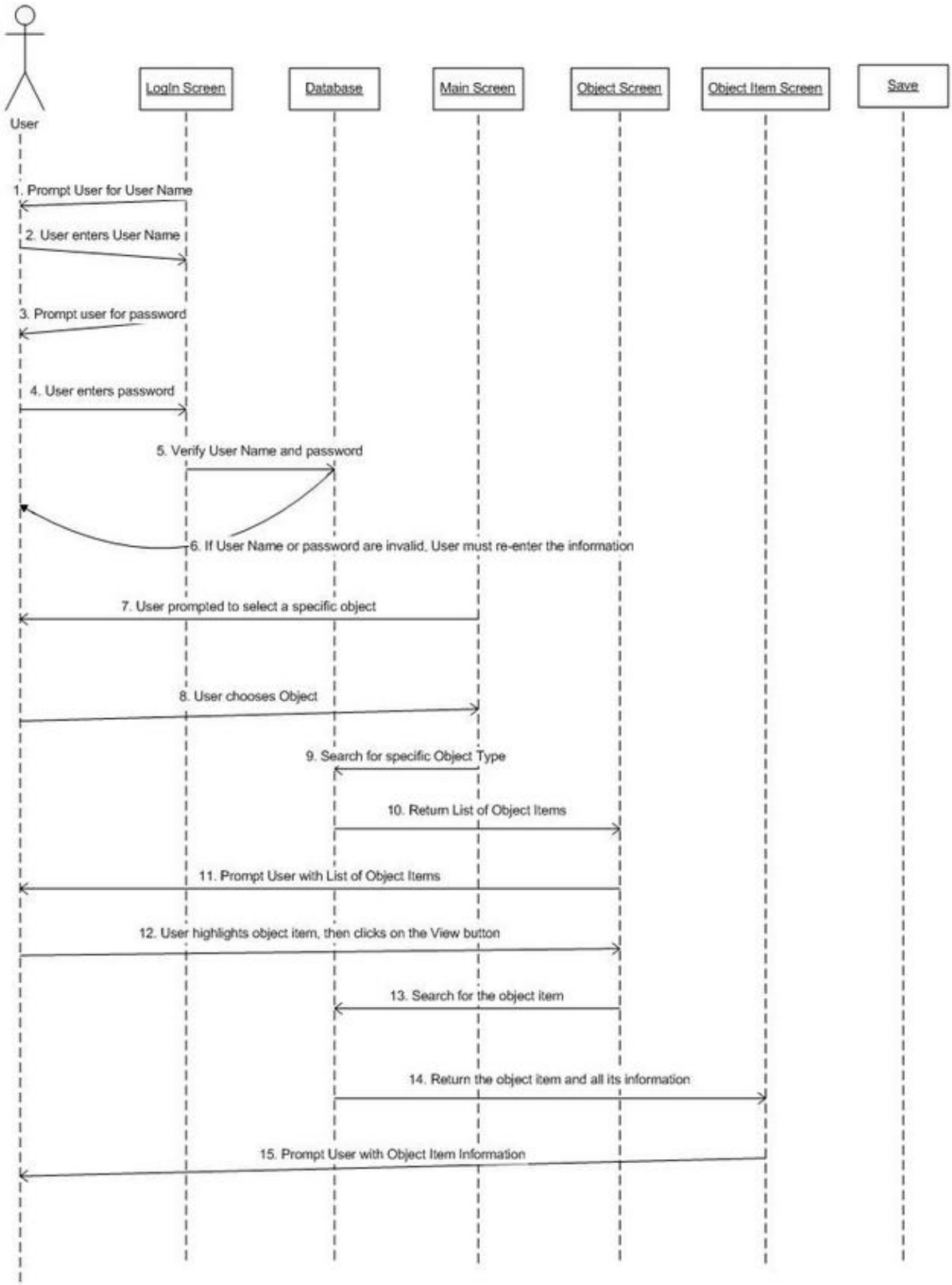
Pre−Configured Reports Sequence Diagram

The user wants to create a Pre−Configured report using data stored in the tracking tool database. The interaction between the user, tracking tool interface, and the database are described in the following diagram.


User
LogIn Screen
Database
Main Screen
Object Screen
Report Screen
View
1. Prompt User for User Name
2. User enters User Name
3. Prompt user for password
4. User enters password
5. Verify User Name and password
6. If User Name or password are invalid, User must re-enter the information
7. User prompted to select a specific object
8. User chooses Report
9. Search for Report Types
10. Return List of Report Types
11. Prompt User with List of Reports and Options
12. User picks an option under the Pre-Configured Report Section, then clicks on the View button
13. Search for the report type
14. Return the Report Type
15. Prompt User with Report Information

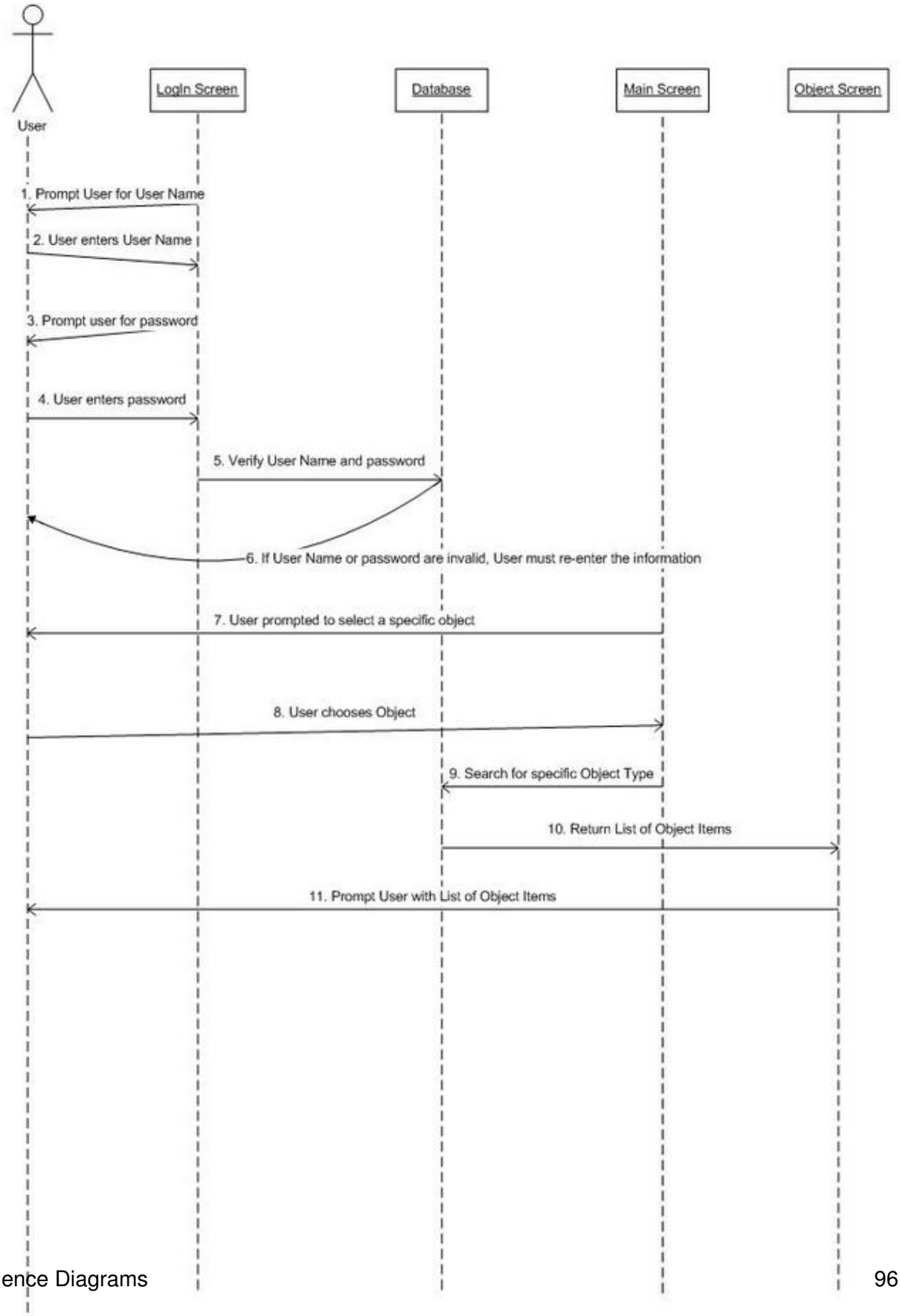
View Object Item Sequence Diagram

The user wants to view an object in the tracking tool database. The interaction between the user, tracking tool interface, and the database are described in the following diagram.


User
LogIn Screen
Database
Main Screen
Object Screen
Object Item Screen
Save
1. Prompt User for User Name
2. User enters User Name
3. Prompt user for password
4. User enters password
5. Verify User Name and password
6. If User Name or password are invalid, User must re-enter the information
7. User prompted to select a specific object
8. User chooses Object
9. Search for specific Object Type
10. Return List of Object Items
11. Prompt User with List of Object Items
12. User highlights object item, then clicks on the View button
13. Search for the object item
14. Return the object item and all its information
15. Prompt User with Object Item Information

View Object Item List Sequence Diagram

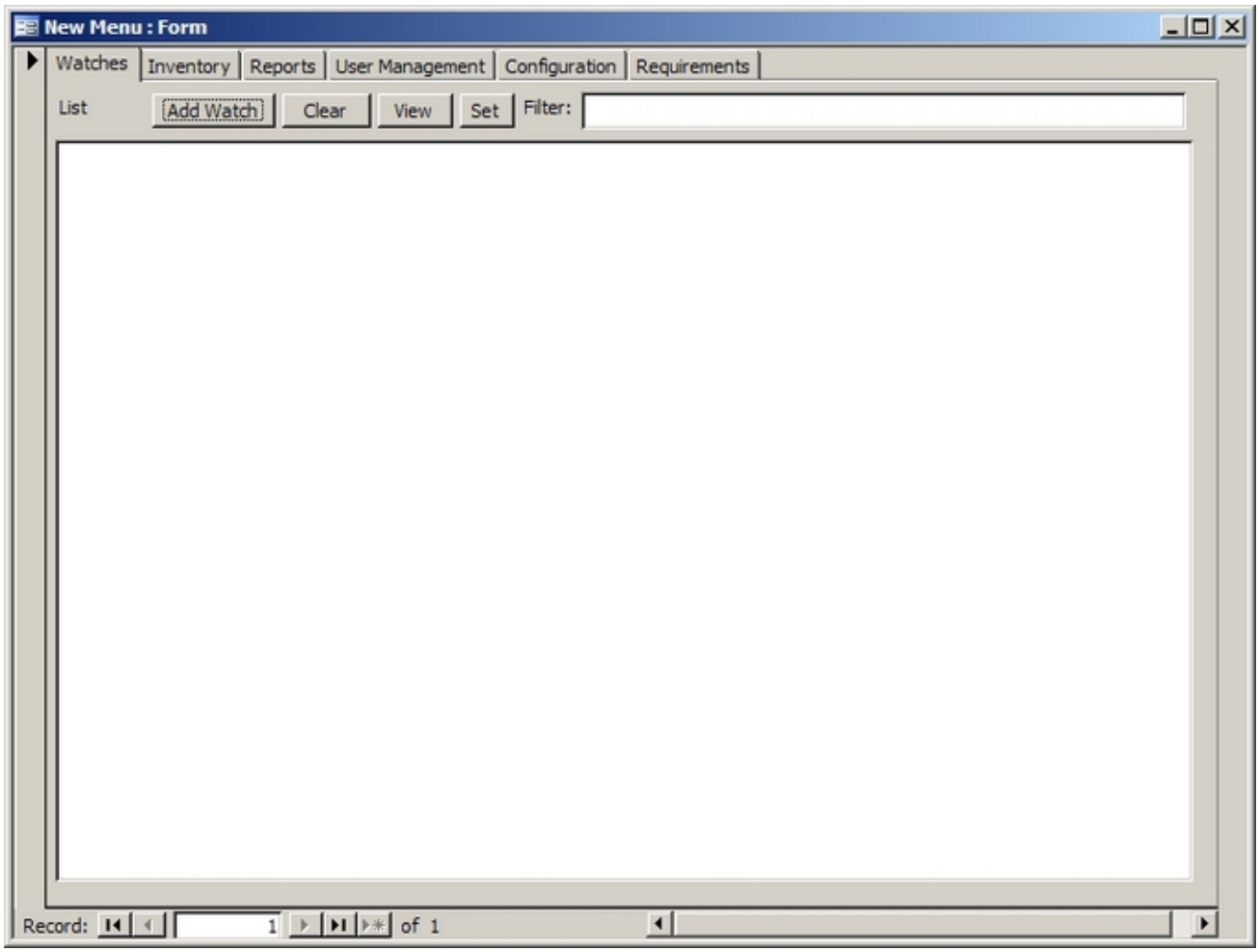
The user wants to view all the objects (of a specific type) in the tracking tool database. The interaction between the user, tracking tool interface, and the database are described in the following diagram.


User
LogIn Screen
Database
Main Screen
Object Screen
1. Prompt User for User Name
2. User enters User Name
3. Prompt user for password
4. User enters password
5. Verify User Name and password
6. If User Name or password are invalid, User must re-enter the information
7. User prompted to select a specific object
8. User chooses Object
9. Search for specific Object Type
10. Return List of Object Items
11. Prompt User with List of Object Items

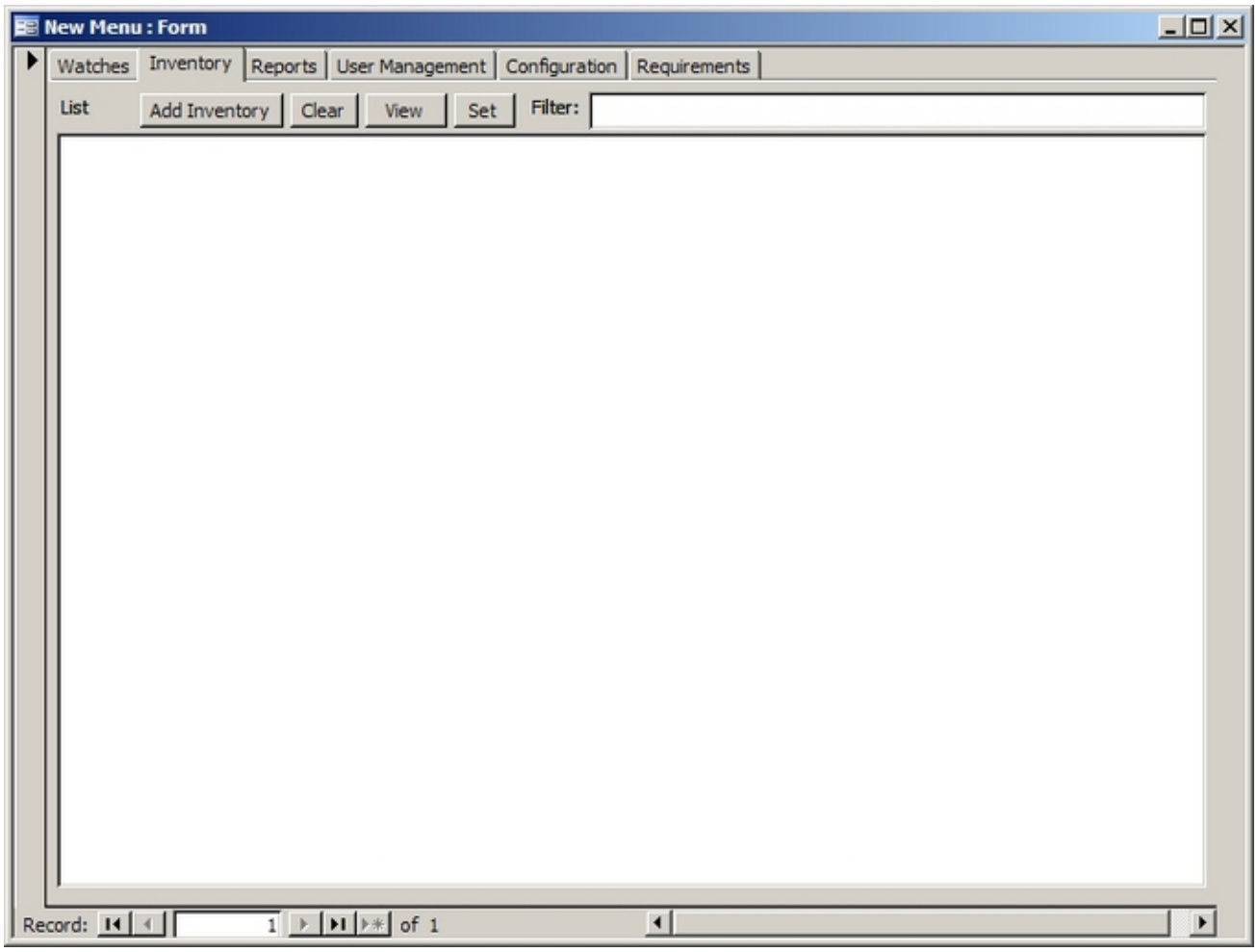
6 User interface Design

6.1 Description of the user interface

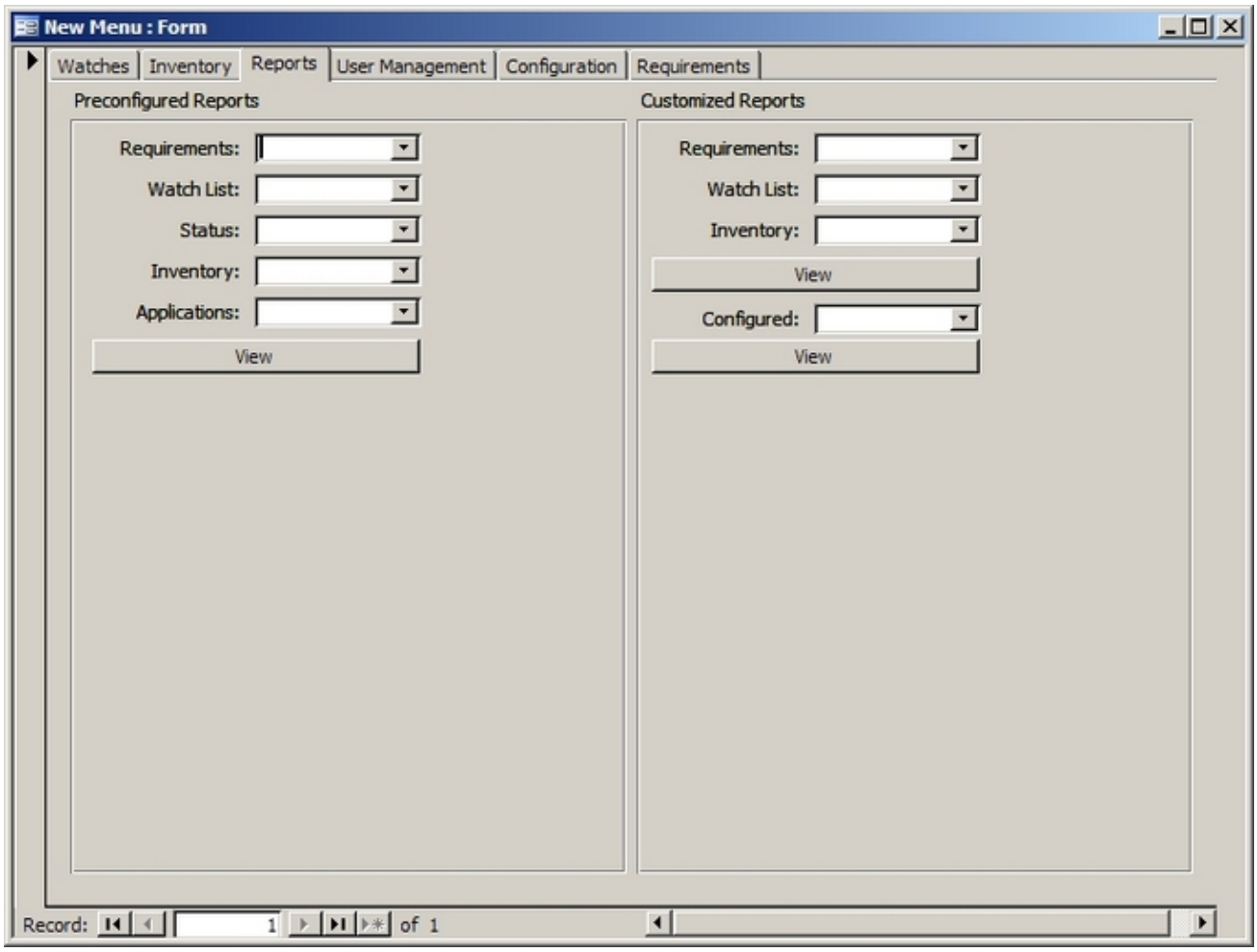
6.1.1 Watches



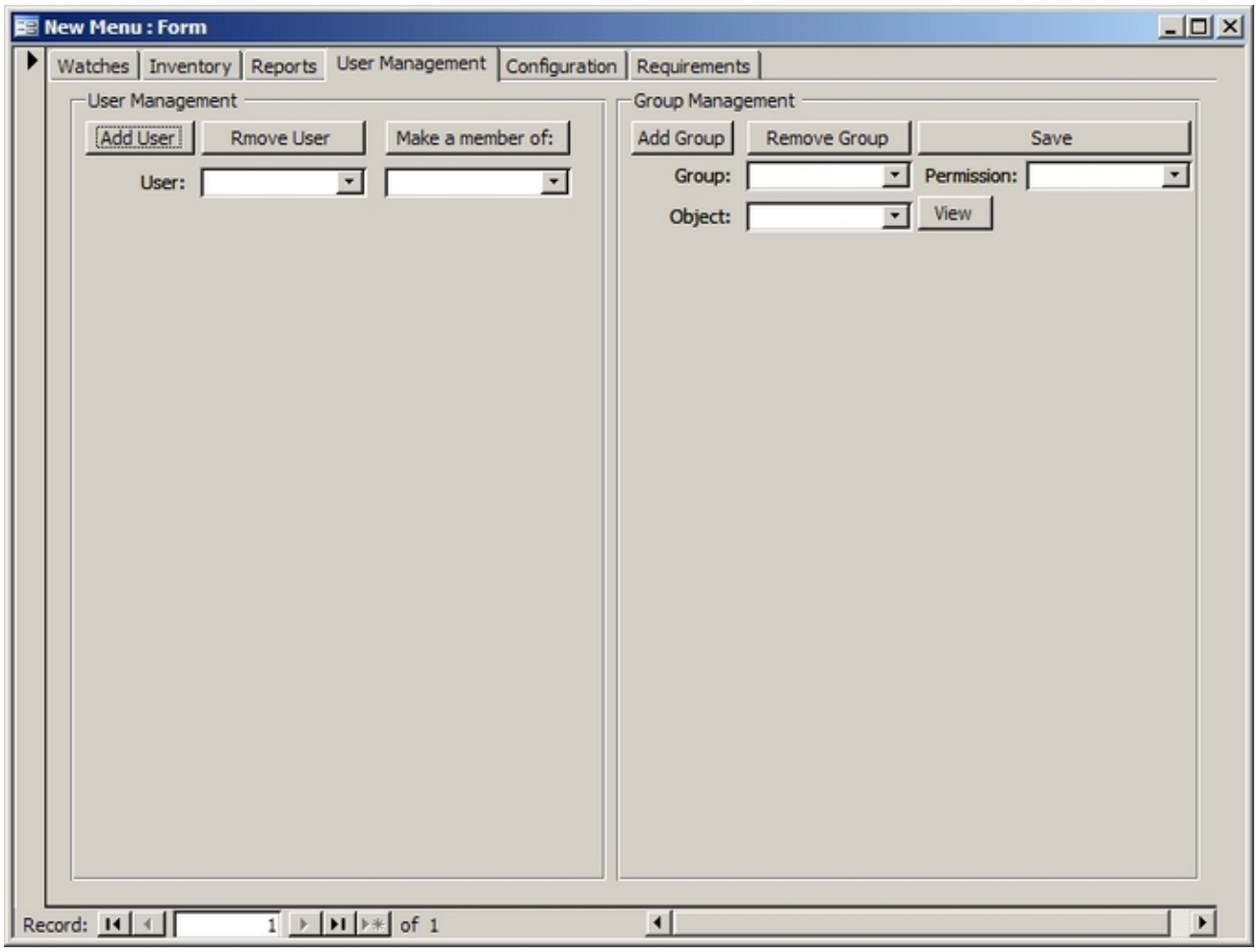
6.1.2 Inventory


New Menu : Form
Watches
Inventory
Reports
User Management
Configuration
Requirements
List
Add Inventory
Clear
View
Set
Filter:
Record:
1
of 1

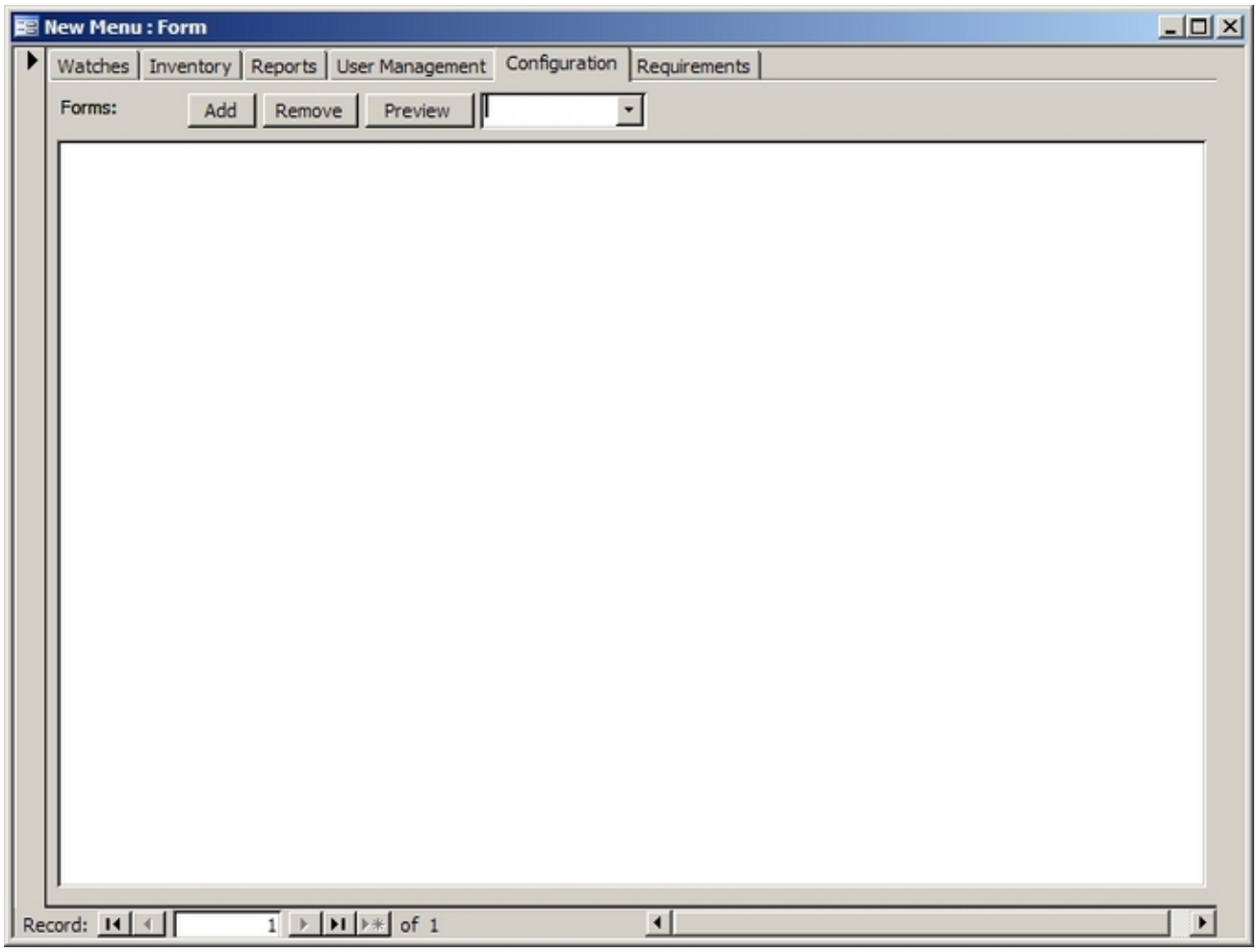
6.1.3 Reports



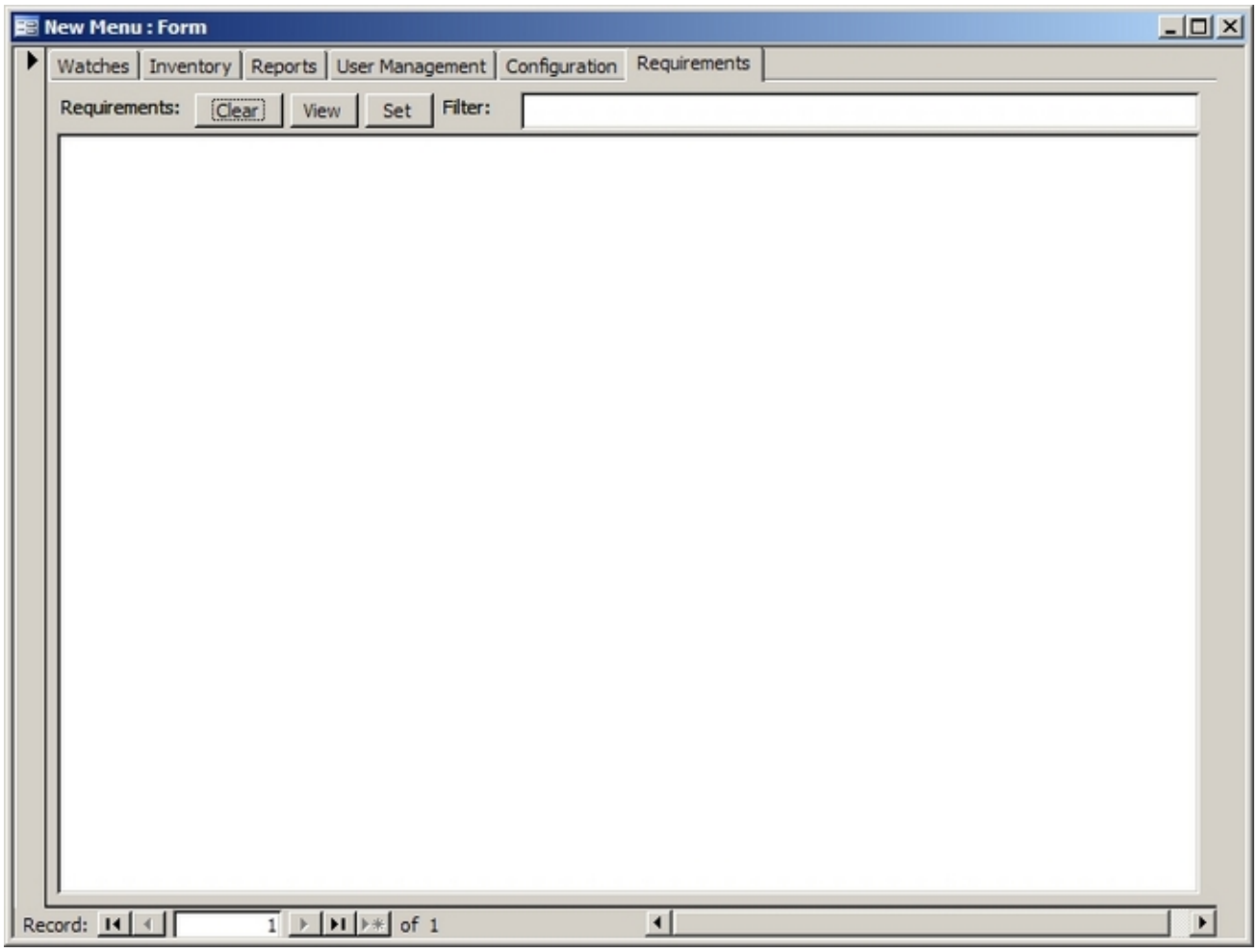
6.1.4 User Management



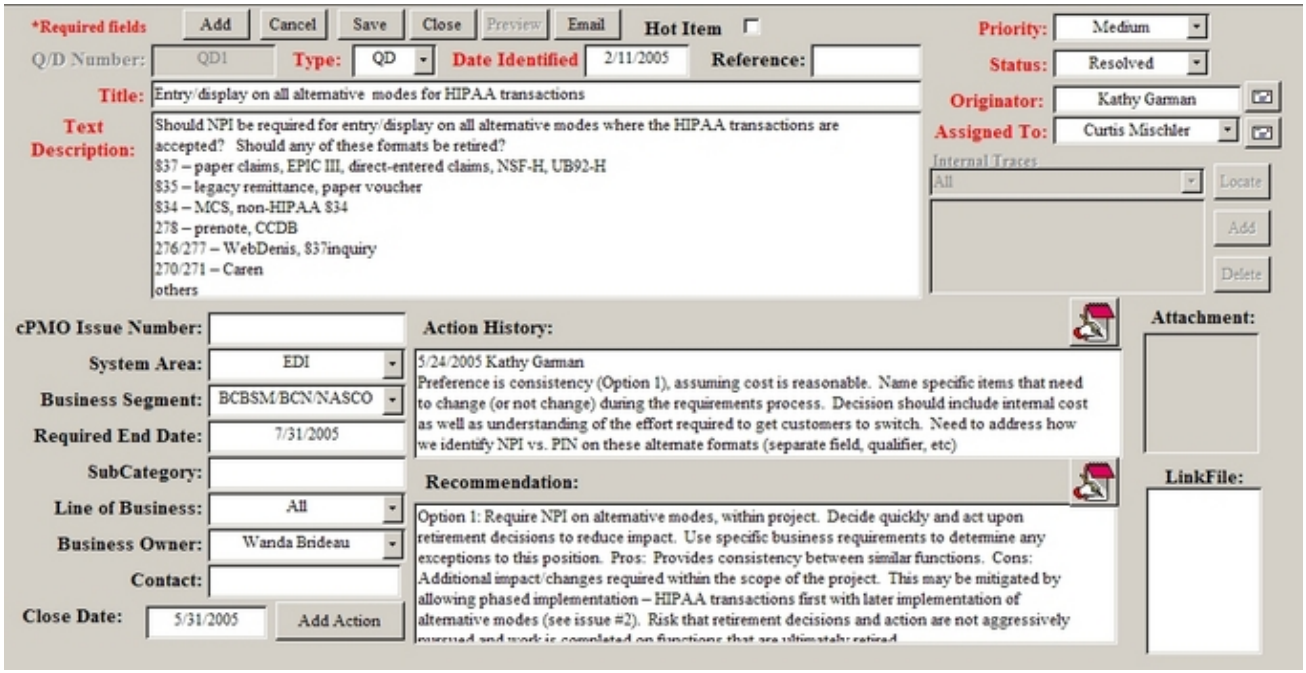
6.1.5 Configuration



6.1.6 Requirements



6.1.7 Generic Add Window


*Required fields
Add
Cancel
Save
Close
Preview
Email
Hot Item
Priority:
Medium
Q/D Number:
QD1
Type:
QD
Date Identified
2/11/2005
Reference:
Status:
Resolved
Title:
Entry/display on all alternative modes for HIPAA transactions
Originator:
Kathy Garman
Text Description:
Should NPI be required for entry/display on all alternative modes where the HIPAA transactions are accepted? Should any of these formats be retired?
$37 – paper claims, EPIC III, direct-entered claims, NSF-H, UB92-H
$35 – legacy remittance, paper voucher
$34 – MCS, non-HIPAA $34
278 – prenote, CCDB
276/277 – WebDenis, 837inquiry
270/271 – Caren
others
Assigned To:
Curtis Mischler
Internal Traces
All
Locate
Add
Delete
cPMO Issue Number:
Action History:
Attachment:
System Area:
EDI
5/24/2005 Kathy Garman
Preference is consistency (Option 1), assuming cost is reasonable. Name specific items that need to change (or not change) during the requirements process. Decision should include internal cost as well as understanding of the effort required to get customers to switch. Need to address how we identify NPI vs. PIN on these alternate formats (separate field, qualifier, etc)
Business Segment:
BCBSM/BCN/NASCO
Required End Date:
7/31/2005
SubCategory:
Recommendation:
LinkFile:
Line of Business:
All
Option 1: Require NPI on alternative modes, within project. Decide quickly and act upon retirement decisions to reduce impact. Use specific business requirements to determine any exceptions to this position. Pros: Provides consistency between similar functions. Cons: Additional impact/changes required within the scope of the project. This may be mitigated by allowing phased implementation – HIPAA transactions first with later implementation of alternative modes (see issue #2). Risk that retirement decisions and action are not aggressively
Business Owner:
Wanda Brideau
Contact:
Close Date:
5/31/2005
Add Action

6.2 Objects and actions

6.2.1 Watches

The watch screen consists of four buttons just below the row of tabs. The first button on the left is the add watch button. When the user clicks this button, the add widow will launch. The next button to the left is the view button. If the user has selected a watch item from the list below the button, this will cause the watch item to launch in the add window. The next two buttons deal with the text field that is all the way to the right of the window. Entering text into this box will allow the user to sort though the items in the list below. When the user has entered some qualifiers into the box and presses the set button, the filter will be set and only matching items will appear. The other button in this set is the clear button and it clears the text field so the user can see all the original entries or start over with a new filter string.

6.2.2 Inventory

The inventory screen consists of four buttons just below the row of tabs. The first button on the left is the add inventory button. When the user clicks this button, the add widow will launch. The next button to the left is the view button. If the user has selected an inventory item from the list below the button, this will cause the inventory item to launch in the add window. The next two buttons deal with the text field that is all the way to the right of the window. Entering text into this box will allow the user to sort though the items in the list below. When the user has entered some qualifiers into the box and presses the set button, the filter will be set and only matching items will appear. The other button in this set is the clear button and it clears the text field so the user can see all the original entries or start over with a new filter string.

6.2.3 Reports

The reports window is used to print out reports that are either stored in the program or created by the user in one of two ways. To print out a preconfigured report, the user would use the section of controls in the left gutter. The user can select from the five report types in the drop down boxes and select sub types of those. After which the user can press the View button just under the drop down boxes to see the report filled in with data from the projects. The right gutter contains the controls to generate custom reports. The first option for custom report generation is a simple three column report that is filled in with what the user selects from the three drop down boxes in the top center portion of the right gutter. After the user makes a selection from the boxes, if they press the View button directly below, they will see a report generated for them. The second way to have user select a user designed custom report from the drop down box below the first View button. After the user selects a report, they press the View button below that to have the report generated for them.

6.2.4 User Management

The user management screen consists of two gutters that separate the functionality of this form. The right gutter is user to manage users. The first on the left, Add User, causes a popup window to open with a text enter box inside and an ok and cancel buttons. The user can enter a new user name and if it does not exist in the program already, then the user will be added and the popup will close. If the user is already in the system, then the user will have to enter a new name or click the cancel button. When a user is in the system, they will show up in the user drop box. If a user is selected and the Remove user button is selected, then that user will be removed from the system. A user can be added to a group by selecting the user and the group in the drop down to the right of the user name, and then pressing the "Make a member of:" button. This will add a user to the selected group. The right gutter is devoted to group management. The first on the left, Add Group, causes a popup window to open with a text enter box inside and an ok and cancel buttons. The user can enter a group

name and if it does not exist in the program already, then the group will be added and the popup will close. If the group is already in the system, then the user will have to enter a new name or click the cancel button. The next button is the Remove group button. If this button is pressed while a group is selected, then that group will be removed from the system. The Save button is used to save permissions for groups. When a group is selected and the Permission and Object dropdowns have items selected in them, then pressing the save button will assign them the permission for the object. Lastly, the View button will show a list of the groups and the permission and user associated with them.

6.2.5 Configuration

The configuration window has three buttons and a drop down. This screen is used to list and preview custom reports that have been created. The first button on the left called Add will launch the form designer for the user with a default template. The next two buttons will work in conjunction with the drop down box to the right of them. The Remove button will remove the selected form from the dropdown, while the Preview button will display the form in the space below the buttons

6.2.6 Requirements

The requirements screen consists of three buttons just below the row of tabs. The first button to the left is the View button. If the user has selected a requirement item from the list below the button, they will cause the requirement to launch in the add window. The next two buttons deal with the text field that is all the way to the right of the window. Entering text into this box will allow the user to sort though the items in the list below. When the user has entered some qualifiers into the box and presses the set button, the filter will be set and only matching items will appear. The other button in this set is the clear button and it clears the text field so the user can see all the original entries or start over with a new filter string.

6.2.7 Generic Add Window

Starting at the top and working from left to right most of the original buttons from the "Add" window still exist. First the "Cancel" button will close the window with out saving any of the information entered to the data base. It leaves the information as it was or the new ticket uncommitted. If the user wants to save the ticket that they have made or the changes to it then they would click the "Save" button. If all the required fields are filled out then clicking the "Save" button will save the information written to the fields and close the window. If a user is unsure of how the object will look before they save it then they can click on the "Preview" button. This launches what is entered in the data fields in a report, so show the user a preview it before submitting it to the data base. If the user is unhappy with what is displayed then they can change it or cancel the creation of the object. Finally for the top row there is the "email" button. "Email" gives the user the option to mail the object to someone. When clicked, the program will forward the preview to there mail client. For the objects below the first row of buttons from the left to right there are; type, date identified, priority, status, originator, and assigned to. The "Type" drop down identifies what type of item it will be. "Data Identified" is the date that the ticket was created though this is editable by the user. Then the "Priority" drop down allows the user to select from the predefined priorities and assign them to the object. "Status" allows the user to assign the status of the object from the drop down with predefined status classes. Moving just below status is the "Originator" and "Assigned to" fields. The originator is the creator of the object, while the assignee is the person who is given the object. The assignee can be changed and both assignee and originator can be sent messages though email from the program. This can be done by selecting the "Email" buttons directly to the right of the fields. Now again starting at the left below the last set of controls and working down. "Title" is the first field. This is where the title of the object goes. A description of the object goes in the "Text Description" field and along with title this is a required field. Next is the "System area" identifies what area of the system is dose this object refers to. "Required end date" is the date that the object is required to be

completed for the business that is described in "Business owner." The user also can find the contact with that business though the field for "Contact." Last in this column of controls is the "Close date" field. This shows the date that the tick was closed if the ticket was closed. Now in the internal controls in the window are the Action history and Recommendation text areas, both editable, and the attachments and files box. To edit the Recommendations and Action history text areas click on the notepad icons to the right. Clicking on the icons will launch a window with a text field that the user can type in. The user can then choose to save the changes or cancel with out appending them to the list. The last item on the form is the internal traces box, this show links between items in the database. Users can add and remove links as well as see a from with all the links to the object.

6.3 Interface design rules

The focus for the components in the user interface is simplicity, common location, and close proximity. The focus on proximity was considered as a way to keep all the items that the user was using close to each other so the eyes would not have to wander the screen area to find what they were looking for. Essentially, this makes the total search area smaller so that searching for the desired command will not take as long. The next approach used was to keep as many of the menus as similar as possible though the interface. The intention was to build on what the user knew about one window and allow them the ability to use it on the next. This was not possible on all the windows. In those cases, similar selection methods were used with the same intent. Lastly, simplicity was considered important since we want the tool to be as user friendly as possible. The second reason was because the tool is supposed to be a generic version of the tool used by NPI. If the tool was supposed to be made universally applicable, it seems that the user interface should too. The first thing that was used to achieve more simplicity was to diverge from a main menu where users selected what function of the tool they wanted to use, to a main form that had tabs with all the primary functionality of the tool listed on each. From here, users could access the same types of functionality in a much cleaner environment. The other step to maintain simplicity was to keep as much of the content on each of these tabs as possible.

6.4 UIDS description

The system used to design the UI is the form builder in Microsoft Access 2003.

7 Restrictions, limitations, and constraints

1. The tracking tool database must be able to operate on a Windows machine.
2. Microsoft Access must be used as the tool to develop the user interface.
3. A SQL server is currently being used at BCBSM.
4. The deadline for software completion is December 20, 2005.
5. The tracking tool database will consist of at most 1000 records.

8 Testing Process

8.1 Classes of tests

The following classes of tests will be used on the tracking tool:

1. *Unit Testing:*
The goal of Unit Testing is the verificaton of each module of the tracking tool. Its focus is on the internal processing logic and data structures within a module. Some examples of test cases that we plan on having in the test plan are: making sure an item is actually added to the database, or making sure an item is actually deleted from the database.
2. *Integration Testing:*
Integration Testing verifies that once the tracking tool is integrated, that all modules work properly together and that overall system performance is achieved. Some examples of test cases that we plan on having in the test plan are: adding an item to the database then making sure you can view the item on the object list, and making sure the login function works after the add watch item function is added.
3. *System Testing:*
The purpose behind System Testing is to fully exercise the computer−based system.
The types of System Tests we will conduct are:

- Security Testing: This is a series of test that verify that security features built into the tracking tool do perform and protect. Some examples of test cases that we plan on having in the test plan are: using an invalid name and making sure the user is not allowed to access the tool, and given a valid username and password, making sure the user can only access the areas of the tool where they have proper privileges.
- Performance Testing: This is a series of tests that verify the performance and speed of the tracking tool. Some examples of test cases that we plan on having in the test plan are: making sure the user can add an item to the database within 3 seconds, and the same for delete an item.

4. *Validation Testing:*
Validation Testing begins when Integration Testing ends, and the tracking tool is completely assembled. The goal is to validate that the tracking tool operates in the manner that the client requested. Some examples of test cases that we plan on having in the test plan are: user can log into the tool, and user can generate reports.

8.2 Performance bounds

Performance Requirements include:

1. Handling of at most 50 multiple users at one time.
2. Tracking Tool should be take no more than 3 seconds to store an object.
3. Tracking Tool should be take no more than 3 seconds to retrieve an object.
4. Tracking Tool should be take no more than 10 seconds to create a report.

8.3 Identification of critical components

The following situations are critical and demand special attention during testing:

1. Confirmation of a valid username.
2. Confirmation of a valid password.
3. Security of the tracking tool especially the modify, add, view, and delete privileges.
4. Error handling of invalid data.
5. Adding objects to the tracking tool.

9 Appendices

9.1 Requirements traceability matrix

The following table shows how the use cases map back to the requirements for the tracking tool:

Requirement:	1	2	3	4	5	6	7	8	9	10	11	12	13
User Story:													
1							X			X			
2							X			X			
3							X			X			
4							X			X			
5							X			X			
6							X			X			
7							X			X			
8							X			X			
9							X			X			
10							X			X			
11							X			X	X		
12							X			X	X		
13							X			X	X		
14	X						X		X	X	X		
15	X						X		X	X			
16	X						X		X	X			
17	X						X		X	X			
18	X						X		X	X			
19	X						X		X	X			
20	X						X		X	X			
21	X						X		X	X			
22	X						X			X			
23	X						X			X			
24	X						X			X			
25	X						X			X			
26	X						X			X			
27	X						X			X			
28	X						X			X			
29	X						X			X			
30		X	X				X			X			
31		X	X				X			X			
32							X			X			
33	X						X	X		X			

| | | |

35. Make Ad–Hoc Report
36. View Pre–Configured Reports
37. View Report List
38. View Report Object
39. Attach External Object
40. Link Internal Object
41. Link External Object
42. Email To Assigned To Person of a Watch Object
43. Email To Assigned To Person of an Inventory Object
44. Change Log for a Watch Object
45. Change Log for an Inventory Object

The Requirements Legend is as follows:
1. The tool ***must*** provide the ability to track at least 2 distinct classes of project information:
A. ProjectWatches
B. ProjectInventory
2. The tool ***may*** provide a separate ProjectRequirements tracking module.
3. The ProjectRequirements module ***may*** be a read–only view of existing Requisitepro data.
4. The tool ***must*** provide the ability to "link" or otherwise create references between items in the tracking modules.
5. The tool ***must*** provide the ability to "link" or otherwise create references to files, documents, or other artifacts outside the tracking modules.
6. The tool ***must*** provide the ability to "attach" or otherwise embed files, documents, or other artifacts to items in the tracking modules.
7. The tool ***must*** provide basic access security.
A. The tool ***should*** provide a group or role based security model.
B. The tool ***must*** support the following privileges:
* ViewPrivilege
* AddPrivilege
* ModifyPrivilege
* DeletePrivilege
8. The tool ***must*** provide a ReportingModule.
9. All CoreModules ***should*** be configurable.
Configurable attributes ***must*** have the ability to be flagged as required attributes.
Configurable attributes ***must*** have at least the ability to be renamed.
The tool ***must*** provide user the ability to define content of drop down menus.
The tool ***may*** provide a method for configuring the layout of data entry screens.
10. The tool ***must*** provide a GraphicalUserInterface.
11. The tool ***must*** provide a system for tracking user activity.
This activity tracking system ***may*** only track logins and logouts.
12. The tool ***must*** provide a way to email to the originator or to the assigned to person of an object.
13. The tool ***must*** provide a way to log changes made to items tracked by the tool.

9.2 Packaging and installation issues

1. To aid in the installation of the tracking tool for new instances, the team will provide an Administration Manual to the NPI team.
2. Migration Tool instructions will be created to aid the NPI team in the detailed process of moving the current data used by the NPI tracking tool to the generic storage data tables.

3. To aid all new users with using the tool, the team will provide an Tracking Tool End User Manual to BCBSM.

9.3 Design metrics to be used

The team will use architectural design metrics for the tracking tool project. These metrics focus on characteristics of the program architecture with emphasis of the effectiveness of the modules without having to know the inner workings of the module.

The three software complexity measures are:
1. Structural Complexity:
This is calculated by S(i)=f out(i), where f out is the number of modules that are directly invoked by module i.
2. Data Complexity:
This is calculated by D(i)=v(i)/(f out(i) +1), where v(i) is the number of input and output variables that are passed to and from module i.
3. System Complexity:
This is calculated by C(i)=S(i) + D(i).

The following table shows the complexity measures for the 45 use case stories. Please use the use case story legend in the above Tracability matrix section for the following table.

Use Case Story Number(i)	***S(i)***	***D(i)***	***C(i)***
1	3	1.5	4.5
2	3	1.25	4.25
3	1	1	2
4	1	1	2
5	2	2	4
6	3	2.5	5.5
7	3	2	5
8	1	1	2
9	1	1	2
10	2	2	4
11	1	1.5	2.5
12	1	2.5	3.5
13	1	2.5	3.5
14	3	2.5	5.5
15	3	2.5	5.5
16	3	2.5	5.5
17	1	1.5	2.5
18	1	1.5	2.5
19	2	1.33	3.33
20	2	1.33	3.33
21	3	5.5	8.5
22	3	5.25	8.25
23	1	2	3

24	1	2.5	3.5
25	2	1	3
26	3	4.67	7.67
27	3	3.25	6.25
28	1	2	3
29	1	2.5	3.5
30	2	1	3
31	1	2	3
32	1	1.5	2.5
33	1	1	2
34	1	3	4
35	1	2.5	3.5
36	1	1	2
37	1	1	2
38	2	1.33	3.33
39	1	2.5	3.5
40	1	2	3
41	1	2	3
42	1	2.5	3.5
43	1	2.5	3.5
44	1	3	4
45	1	3	4

From the above table, the modules with the most complexity are:
1. Modify an Inventory Item
2. Add an Inventory Item
3. Modify a Watch Item
4. Add a Watch Item

Extra time will be planned into the development and coding of these sections.

9.4 Supplementary information

Pressman, Roger 2005, Software Engineering: A Practitioner's Approach, McGraw–Hill, New York

Adaptable Process Model: Document Templates, 2001 Available at:
http://www.rspa.com/docs/Designspec.html

10 Project Glossary

Here is a collection of terms used throughout the project:

- CoreModules
- GraphicalUserInterface
- MigrationToolManual
- ObjectLinkingAndEmbedding
- PaperPrototyping
- Privileges
    - AddPrivilege
    - DeletePrivilege
    - ModifyPrivilege
    - ViewPrivilege
- ProjectInventory
- ProjectRequirements
- ProjectWatches
- ProjectVelocity
- ReportingModule
- UserStories
- TheCustomer
- TheTeam
- UserGroup

10.1 Core Modules

Core modules are the main modules that the user will interact with. Currently these are the design items filter and the main screen along with the watch list. The functionality of these menus will remain while the exact layout and names will change. It is intended that the names will be customizable to the specific work methodology for the group.

10.2 The Graphical User Interface

The user interface will resemble the current interface fairly close. The intent of this is to capitalize on the experience the user already has. Keeping functional areas of the interface the same will coincide with the expectations the user has for that screen and the application. If the project is taken to another platform then the design of the new forms will be made to closely resemble the old. The only exception to this is if changing the from will increase ease of use in the short term and long term. In such a case a usability proof will be provided for the cause for the change. Until such a decision, the user interface will look like the original screenshots obtained for the group.

10.3 The Migration Tool Manual

The migration tool instructions will be a detailed process to move the current data used by the NPI tracking tool to the generic storage data tables. Since the migration tool is a single use piece of software, it may not

necessarily be a polished application, and could possibly consist only of a manual script for converting the data.

10.4 Object Linking and Embedding

In order to make tracking related objects simpler, the tool will provide the capability to either link to them or embed them in a tracked object. When an object is linked to another object, there is just a simple reference to the linked object that the user can follow to find it. Linked objects may include both other tracking objects, as well as external objects stored on file servers like documents and pictures.

In contrast to linking, embedding an object actually stores it in the database alongside the object it is embedded in. Embedding is very similar to attaching a document to an email. Embedding can only be used on external objects, and should be reserved for small objects like emails or short documents. Larger objects should be linked instead, in order to save space in the database.

10.5 Paper Prototyping

Paper prototyping is a method for rapidly developing a graphical user interface. Basically, the developers and the customer sit down together and sketch, on paper, a mockup of the user interface. The customer then role−plays using the interface to see if the mockup works well. As the prototyping session continues, the user interface will be repeatedly refined until a final, usable interface is developed. Paper prototyping saves greatly on development cost, since the prototypes are never built in actual code, and so can be more quickly and easily redesigned.

10.6 Add Privilege

Users with the "add" privilege for a module may add new objects to it.

10.7 Delete Privilege

Users with the "delete" privilege for a module may delete items within that module.

10.8 Modify Privilege

Users with the "modify" privilege for a module may modify the attributes of an existing object within that module.

10.9 View Privilege

Users with the "view" privilege for a module may view the contents of the module.

10.10 Project Inventory

Project Inventory are the various documents, artifacts or "widgets" for the project. These items may include:

- Detailed designs
- Conceptual designs
- Source code
- Executable components
- Major threads of development

Each of these items may have its own InventoryType that specifies what kind of information needs to be tracked for that item.

10.11 Project Requirements

This module tracks all information that is related to the current project's requirements. Because of The customer's corporate requirement to use RequisitePro to manage requirements, this module is essentially a read−only view of the existing RequisitePro data. It's really only here, so that links can be created from requirements to other tracked items.

10.12 Project Watches

Project Watches are various items which need special attention from the project team. Examples of Watches that exist in the current NPI Tracking Tool are:

- Questions and Decisions
- Related Projects
- Business Readiness
- Multiple PINs

Other projects may define their Watches according to their own particular requirements

10.13 Project Velocity

Project Velocity is the core metric used in Extreme Programming to measure productivity. It is calculated by totalling the number of StoryPoints completed in an iteration. TheCustomer is not permitted to select more than the Project Velocity in UserStories to complete in an iteration. This restriction helps to control scope and keep TheTeam from taking on more work than they can handle for an iteration.

10.14 Reporting Module

The reporting module is where the user will be able to see reports based on queries. Some of these reports will be based on predefined queries and will be called "Predefined reports." The other reports will be made by custom queries and considered "Custom Reports."

10.15 User Stories

User stories are Extreme Programming's answer for requirements gathering. Basically a user story is just that – a story about how the user uses the system, or how the system interacts with other systems. A user story should be a piece of functionality that can be implemented in a single iteration. If it's too big, then it needs to be split into multiple stories. Once a story has been written, it can be assigned to a developer and built.

The way we are using user stories is to first create a large use case. The individual use cases are extremely general. From the general use case come specializations. These specializations, if small enough, may be single user stories, or may need to be split even more. Once the user stories for the use cases are written, they are entered into the ticket tracking system as tickets, and assigned to a developer. The developer then builds the story and tests it. When it passes its unit tests, the story is released to the customer for acceptance testing.

10.16 The Customer

10.16.1 Blue Cross/Blue Shield of Michigan

The tool is being produced for the Special Projects department of BCBSM. This department manages and develops large projects for BCBSM's core systems. We are specifically working with the NPI project to convert their tracking tool into a more general tool that can be used for other projects in the department.

10.16.2 Primary Customer Contact

Polakampalli, Narendra
npolakampalli@bcbsm.com

10.17 The Team

TeamBC are Christopher Smith, Joey Granz, Greg Symons and Amy Hedges. This project is being completed as part of the requirements for the University of Michigan – Dearborn Department of Computer and Information Science's Senior Design (CIS 495) class. This class is intended as a capstone project for undergraduate students in the Computer Science, Information Science, and Software Engineering programs.

Christopher Smith
(313) 610–6092
smithswe@umd.umich.edu

Joey Granz
(734) 624–1685
joeygranz@yahoo.com

Greg Symons
(734) 649–6303
gsymons@umd.umich.edu

Amy Hedges
(313) 310–0080

arhedges@umd.umich.edu

10.18 User Group

A user group is a list of users that share the same privileges. Administration becomes less difficult when privileges are stored on the groups, rather than each user individually. User groups can also be used to define organizational units of a project. For example, all the Business Owners on the project can be assigned to the Business Owners group. Then the tool can query that group's membership to populate a drop down list of user names on a tracking record.

11 Changelog

12

12.1 Changes for Build 3:

12.1.1 Changes for page: DataDesign

- (11/10/05 20:04:17 by gsymons): *fixed page layout table so it looks more like all the other tables.*
- (11/10/05 01:00:04 by gsymons): *Fleshed out the actual design description.*
- (11/09/05 22:56:03 by gsymons): *Added paged ERD.*
- (11/09/05 17:57:15 by arhedges): *spelling corrections*

12.1.2 Changes for page: ComponentLevelDesign

- (11/09/05 23:17:10 by jgranz): *Corrected spelling in View Inventory Object Module*
- (11/09/05 23:12:39 by jgranz): *Corrected grammar in the Change Log Section*
- (11/09/05 18:28:05 by arhedges): *spelling, punctuation, grammar corrections*

12.1.3 Changes for page: SequenceDiagrams

- (11/10/05 00:07:29 by jgranz): *Fixed indentation of sub categories*
- (11/10/05 00:03:13 by jgranz): *Added documentation to the sequence diagrams.*

12.1.4 Changes for page: UserInterfaceDesign

- (11/09/05 21:18:36 by smithswe): *Added descritpion of the "Add Window"*
- (11/09/05 02:01:04 by jgranz): *Corrected spelling and grammar*

12.1.5 Changes for page: TestingIssues

- (11/10/05 13:43:41 by jgranz): *Added example test cases to the test process*
- (11/09/05 01:42:20 by jgranz): *Changed format, and correct spelling*

12.1.6 Changes for page: AppendicesSection

- (11/10/05 23:11:52 by gsymons): *Rotated the traceability matrix so that it doesn't get cut off in the pdf.*
- (11/09/05 01:32:10 by jgranz): *Corrected alignment of paragraph*
- (11/09/05 01:12:34 by jgranz): *Added Design Metrics Table*

12.1.7 Changes to documentation repository items:

- Changeset 84: Autogenerated documentation commit (11/08/05 20:50:04 by)
- Changeset 85: Autogenerated documentation commit (11/08/05 20:50:55 by)
- Changeset 86: Added new images for the ERD. I split them into page sized pieces. (11/09/05 22:40:30 by gsymons)

12.2 Changes for Build 2:

12.2.1 Changes to documentation repository items:

- Changeset 1: Initial repository layout. (10/06/05 10:41:45 by gsymons)
- Changeset 8: Added an images folder. (10/16/05 23:30:36 by gsymons)
- Changeset 9: (10/16/05 23:42:27 by smithswe)
- Changeset 10: Changed mime−type to image/jpeg (10/16/05 23:51:19 by gsymons)
- Changeset 12: (10/17/05 00:09:42 by smithswe)
- Changeset 13: Set mimetypes to image/jpeg (10/17/05 11:13:30 by gsymons)
- Changeset 17: The monkey will kill you, I tell you! Kill you! (10/20/05 11:02:07 by gsymons)
- Changeset 18: (10/20/05 17:03:49 by smithswe)
- Changeset 20: Added favicon (10/21/05 01:55:26 by gsymons)
- Changeset 21: Started on the design model. I've got a rough start of the static object model. (10/27/05 17:56:26 by gsymons)
- Changeset 22: Added some attributes and a bit of refactoring... I think (I'm in Linux at the moment, so I don't totally remember what I did:) (10/28/05 19:04:51 by gsymons)
- Changeset 23: Fleshed out TrackedObjectInstance, WatchInstance, and a few others. (10/29/05 20:29:46 by gsymons)
- Changeset 27: Autogenerated documentation commit (10/30/05 21:27:48 by)
- Changeset 28: Autogenerated documentation commit (10/30/05 21:48:13 by)
- Changeset 29: I think the static model is pretty much done. I now need to do a ERD for the actual O/R mapping. (10/30/05 23:30:28 by gsymons)
- Changeset 30: Started on the ERD. (10/31/05 01:46:51 by gsymons)
- Changeset 31: Well everything on the current version of the object model is now on the erd. Of course none of this includes anything from requirements tracking, because, well, I haven't done anything on requirements tracking yet:) I sorta forgot about it in the Object Model. (10/31/05 22:06:14 by gsymons)
- Changeset 32: add group activity diagram (10/31/05 23:12:26 by arhedges)
- Changeset 34: i messed something up so now i'm fixing it (11/01/05 00:55:37 by arhedges)
- Changeset 37: moved the design model around a bit so that it could be made into an image that didn't greek out all the text. I'm still not convinced it's readable. (11/01/05 01:39:11 by gsymons)
- Changeset 38: added activity diagram for addgroup (11/01/05 01:40:12 by arhedges)
- Changeset 39: added a jpeg of the erd. (11/01/05 01:46:02 by gsymons)
- Changeset 41: adduser activity diagram (11/01/05 01:56:13 by arhedges)
- Changeset 43: Autogenerated documentation commit (11/01/05 02:00:12 by)
- Changeset 44: Autogenerated documentation commit (11/01/05 03:07:09 by)
- Changeset 45: Autogenerated documentation commit (11/01/05 07:53:46 by)
- Changeset 46: (11/01/05 13:44:40 by smithswe)
- Changeset 47: activity diag for viewgroup (11/02/05 01:13:08 by arhedges)
- Changeset 48: Autogenerated documentation commit (11/02/05 02:00:24 by)
- Changeset 50: added activity diagram for addgroup (11/02/05 21:27:38 by arhedges)
- Changeset 51: added activity diagram for addgroup (11/02/05 21:27:54 by arhedges)
- Changeset 52: adduser activity diagram (11/02/05 21:28:08 by arhedges)
- Changeset 53: activity diag for viewgroup (11/02/05 21:28:23 by arhedges)
- Changeset 54: add diagrams (11/02/05 21:30:00 by arhedges)
- Changeset 55: add diagrams (11/03/05 01:20:17 by arhedges)
- Changeset 56: Autogenerated documentation commit (11/03/05 02:00:21 by)
- Changeset 57: (11/03/05 10:43:36 by smithswe)
- Changeset 58: (11/03/05 11:24:34 by smithswe)

- Changeset 59: Here's an update to the ERD. I'm still working on coming up with a decent web−compatible image. (11/03/05 23:52:06 by gsymons)
- Changeset 60: (11/04/05 01:51:36 by jgranz)
- Changeset 61: add diagrams (11/04/05 02:47:08 by arhedges)
- Changeset 62: deleting visio drawing and putting jpg pictures instead (11/04/05 03:13:14 by jgranz)
- Changeset 63: jpg versions of the activity diagrams (11/04/05 03:16:35 by jgranz)
- Changeset 64: had misspelling in image (11/04/05 03:51:41 by jgranz)
- Changeset 65: corrected spelling (11/04/05 03:52:39 by jgranz)
- Changeset 66: adding sequence diagrams (11/04/05 04:10:45 by jgranz)
- Changeset 67: add sequence diagram (11/04/05 04:26:56 by jgranz)
- Changeset 69: (11/06/05 19:34:38 by smithswe)
- Changeset 70: Resized the images to 680 pixels wide so they don't get cropped by HTMLDOC. Also, a windows Thumbs database maded it into the repository somehow. It's gone now. (11/07/05 02:51:44 by gsymons)
- Changeset 72: Added Id to svn:keywords for the html files. (11/07/05 03:17:55 by gsymons)
- Changeset 73: Autogenerated documentation commit (11/07/05 03:20:38 by)
- Changeset 74: Autogenerated documentation commit (11/07/05 03:21:17 by)
- Changeset 75: add diagrams (11/07/05 23:47:26 by arhedges)
- Changeset 76: Added use case map and diagram to repository. (11/08/05 17:08:48 by gsymons)
- Changeset 77: resized images (11/08/05 17:37:19 by gsymons)
- Changeset 79: Autogenerated documentation commit (11/08/05 20:24:02 by)
- Changeset 82: Autogenerated documentation commit (11/08/05 20:33:14 by)
- Changeset 83: Added completion chart to repository (11/08/05 20:45:31 by gsymons)

12.3 Changes for Build 1:

12.3.1 Changes for page: ProjectOverviewAndScope

- (11/01/05 23:45:54 by jgranz): *Removed Major Constaints section per client request*
- (11/01/05 23:40:59 by jgranz): *Move Expected Software Response from Testing Process to Introduction*
- (10/25/05 15:47:34 by smithswe): *Took out "See Also."*
- (10/25/05 15:01:41 by arhedges): *No comment*
- (10/20/05 16:31:08 by arhedges): *No comment*
- (10/12/05 22:45:41 by jgranz): *No comment*
- (10/12/05 22:45:02 by jgranz): *No comment*
- (10/07/05 16:50:14 by gsymons): *No comment*
- (10/07/05 16:04:41 by gsymons): *No comment*

12.3.2 Changes for page: DataDesign

- (11/04/05 03:30:26 by arhedges): *No comment*
- (11/01/05 01:45:12 by gsymons): *No comment*
- (11/01/05 01:43:06 by gsymons): *No comment*
- (11/01/05 01:41:36 by gsymons): *No comment*
- (10/24/05 11:53:51 by jgranz): *No comment*

12.3.3 Changes for page: ComponentLevelDesign

- (11/07/05 23:58:21 by arhedges): *added architectual design section*
- (11/06/05 23:11:08 by gsymons): *fixed smart quotes*
- (11/06/05 23:02:58 by gsymons): *fixed smart quotes*
- (11/04/05 17:12:41 by arhedges): *added components*
- (11/04/05 12:53:42 by arhedges): *No comment*
- (11/04/05 12:45:22 by arhedges): *added pictures*
- (11/04/05 04:32:45 by jgranz): *took out unnecessary sections of pressman template*
- (11/04/05 03:46:28 by jgranz): *Added Act. Diagrams*
- (11/04/05 03:20:56 by arhedges): *added components*
- (11/04/05 00:48:21 by jgranz): *added id to end of Item for Change Log Algorithm*
- (11/04/05 00:39:33 by jgranz): *Correct spelling errors*
- (11/03/05 23:03:34 by jgranz): *Added Inventory and Watch View Oject and View List*
- (11/03/05 21:32:29 by gsymons): *got rid of more latin−1 characters.*
- (11/03/05 21:20:13 by jgranz): *Added Linking, Change Log Use Case Stories & Modified Email To Use Case Stories*
- (11/03/05 21:10:41 by gsymons): *fixed latin−1 characters*
- (11/03/05 01:22:12 by arhedges): *No comment*
- (11/02/05 23:35:47 by jgranz): *Added both Email Use Cases*
- (11/02/05 21:42:26 by arhedges): *No comment*
- (11/02/05 21:29:49 by jgranz): *Added LogOut Module*
- (11/02/05 15:23:46 by arhedges): *fix some typos*
- (11/02/05 15:00:53 by arhedges): *No comment*
- (11/02/05 10:30:30 by smithswe): *Changed page formatting*
- (11/02/05 01:19:00 by arhedges): *No comment*
- (11/02/05 01:16:41 by arhedges): *No comment*
- (11/02/05 00:03:38 by jgranz): *changed quotation marks*
- (11/01/05 20:57:50 by arhedges): *fixed "smart quotes"*
- (11/01/05 01:57:18 by arhedges): *No comment*
- (11/01/05 01:40:47 by arhedges): *No comment*
- (11/01/05 01:04:25 by arhedges): *No comment*
- (11/01/05 01:01:51 by arhedges): *No comment*
- (11/01/05 00:49:40 by arhedges): *No comment*
- (10/31/05 13:50:00 by arhedges): *No comment*
- (10/30/05 23:55:38 by arhedges): *No comment*
- (10/30/05 23:43:04 by arhedges): *No comment*
- (10/30/05 20:01:13 by arhedges): *No comment*
- (10/30/05 19:06:09 by arhedges): *No comment*
- (10/30/05 02:20:34 by arhedges): *No comment*
- (10/30/05 02:14:49 by arhedges): *No comment*
- (10/24/05 11:56:24 by jgranz): *No comment*
- (10/24/05 11:55:46 by jgranz): *No comment*

12.3.4 Changes for page: SequenceDiagrams

- (11/04/05 04:29:47 by jgranz): *Put Images into Subversion*
- (10/31/05 02:01:05 by jgranz): *No comment*

12.3.5 Changes for page: UserInterfaceDesign

- (11/07/05 13:26:50 by smithswe): *Removed special "" characters*
- (11/07/05 13:22:05 by smithswe): *Appended more UI discription*
- (11/04/05 03:29:20 by arhedges): *No comment*
- (11/04/05 00:51:53 by smithswe): *Update and Save 1, Images, Deisgn Rules, and Inviroment*
- (11/03/05 21:41:34 by gsymons): *missed one....*
- (11/03/05 21:39:40 by gsymons): *removed latin−1 characters*
- (11/02/05 10:32:53 by smithswe): *Changed page formatting*
- (11/01/05 14:03:02 by smithswe): *No comment*
- (10/31/05 12:09:42 by smithswe): *No comment*
- (10/30/05 20:02:11 by smithswe): *I put the 'I' in interface*
- (10/24/05 11:57:49 by jgranz): *No comment*

12.3.6 Changes for page: RestrictionsLimitations

- (11/02/05 00:14:58 by jgranz): *Added 1000 record database as discussed in meeting*
- (10/30/05 21:49:10 by jgranz): *No comment*
- (10/29/05 12:58:32 by jgranz): *No comment*
- (10/24/05 11:59:14 by jgranz): *No comment*

12.3.7 Changes for page: TestingIssues

- (11/02/05 00:18:35 by jgranz): *Corrected format issues*
- (11/02/05 00:11:51 by jgranz): *Added Performance bounds discussed in meeting*
- (11/01/05 23:38:57 by jgranz): *No comment*
- (11/01/05 23:38:37 by jgranz): *Changed Issues to Process, move validation after system testing, took out Expected software response (moved to Introduction)*
- (10/30/05 21:49:46 by jgranz): *No comment*
- (10/29/05 12:52:32 by jgranz): *No comment*
- (10/29/05 12:18:41 by jgranz): *No comment*
- (10/24/05 12:01:01 by jgranz): *No comment*

12.3.8 Changes for page: AppendicesSection

- (11/07/05 20:09:07 by jgranz): *Changed matrix−missing requirement 13 and use case stories 42−45 needed to be labeled correctly*
- (11/02/05 00:56:33 by jgranz): *Change traceability table/legend*
- (10/30/05 21:47:59 by jgranz): *No comment*
- (10/30/05 21:35:50 by jgranz): *No comment*
- (10/30/05 21:22:15 by jgranz): *No comment*
- (10/24/05 12:02:45 by jgranz): *No comment*

12.3.9 Changes for page: ProjectGlossary

- (10/25/05 19:15:19 by smithswe): *No comment*
- (10/20/05 01:12:07 by gsymons): *Added glossary entry for user groups*
- (10/17/05 22:04:43 by gsymons): *No comment*
- (10/17/05 21:33:24 by gsymons): *Got everything from the SRS listed on the glossary page*

- (10/17/05 20:46:55 by gsymons): *"issues" changed to "watches"*
- (10/09/05 15:42:48 by gsymons): *No comment*
- (10/09/05 15:41:27 by gsymons): *Added project glossary index*

12.3.10 Changes for page: CoreModules

- (11/01/05 15:07:21 by smithswe): *standardized page layout format*
- (10/20/05 19:41:07 by arhedges): *No comment*
- (10/12/05 18:42:37 by smithswe): *Defined Core Modules*

12.3.11 Changes for page: GraphicalUserInterface

- (11/01/05 15:02:04 by smithswe): *standardized page layout format*
- (10/20/05 19:40:10 by arhedges): *No comment*
- (10/20/05 19:35:29 by arhedges): *No comment*
- (10/12/05 18:09:20 by smithswe): *No comment*
- (10/12/05 18:08:45 by smithswe): *Described the UI and linked to pictusers we allready have*

12.3.12 Changes for page: MigrationToolManual

- (11/03/05 23:10:13 by gsymons): *Cut the Project Glossary part of the heading.*
- (11/01/05 15:08:40 by smithswe): *standardized page layout format*
- (10/20/05 19:39:25 by arhedges): *No comment*
- (10/20/05 19:37:25 by arhedges): *No comment*
- (10/17/05 21:26:32 by gsymons): *No comment*
- (10/12/05 22:58:14 by jgranz): *No comment*
- (10/12/05 17:40:57 by smithswe): *Added disctiption to the "documentataion of the tool" section*

12.3.13 Changes for page: ObjectLinkingAndEmbedding

- (11/01/05 15:05:26 by smithswe): *standardized page layout format*
- (10/25/05 16:06:11 by smithswe): *Removed "See Also."*
- (10/09/05 18:26:01 by gsymons): *Fleshed out the definitions, and added a header and a footer*
- (10/06/05 11:04:05 by smithswe): *Defined link v1*

12.3.14 Changes for page: PaperPrototyping

- (11/03/05 23:16:29 by gsymons): *fixed header*
- (11/03/05 23:10:47 by gsymons): *Cut the Project Glossary part of the heading.*
- (10/25/05 19:15:04 by smithswe): *No comment*
- (10/17/05 21:41:04 by gsymons): *No comment*
- (10/17/05 21:40:38 by gsymons): *No comment*

12.3.15 Changes for page: AddPrivilege

- (11/01/05 15:05:47 by smithswe): *standardized page layout format*
- (10/25/05 16:08:32 by smithswe): *Removed "See Also."*
- (10/20/05 20:08:09 by arhedges): *No comment*
- (10/20/05 20:06:35 by arhedges): *No comment*

- (10/09/05 18:42:08 by gsymons): *added header and footer, moved module stuff to ModuleList*
- (10/06/05 07:41:14 by arhedges): *No comment*
- (10/06/05 07:34:13 by arhedges): *No comment*

12.3.16 Changes for page: DeletePrivilege

- (11/03/05 23:11:15 by gsymons): *Cut the Project Glossary part of the heading.*
- (11/01/05 15:06:31 by smithswe): *standardized page layout format*
- (10/25/05 16:08:18 by smithswe): *Removed "See Also."*
- (10/09/05 19:19:03 by gsymons): *No comment*
- (10/06/05 07:36:05 by arhedges): *No comment*

12.3.17 Changes for page: ModifyPrivilege

- (11/01/05 15:06:18 by smithswe): *standardized page layout format*
- (10/25/05 16:08:26 by smithswe): *Removed "See Also."*
- (10/09/05 19:15:27 by gsymons): *added header and footer.*
- (10/06/05 07:41:59 by arhedges): *No comment*
- (10/06/05 07:35:16 by arhedges): *No comment*

12.3.18 Changes for page: ViewPrivilege

- (11/03/05 23:11:42 by gsymons): *Cut the Project Glossary part of the heading.*
- (11/01/05 15:05:37 by smithswe): *standardized page layout format*
- (10/25/05 16:08:42 by smithswe): *Removed "See Also."*
- (10/09/05 18:31:04 by gsymons): *Added header and footer, fixed grammar*
- (10/06/05 07:32:19 by arhedges): *No comment*

12.3.19 Changes for page: ProjectInventory

- (11/01/05 15:03:37 by smithswe): *standardized page layout format*
- (10/25/05 16:07:31 by smithswe): *Removed "See Also."*
- (10/20/05 19:44:34 by arhedges): *No comment*
- (10/09/05 15:53:13 by gsymons): *Added some more detail, a header, and some cross−reference links.*
- (10/06/05 07:30:24 by arhedges): *No comment*

12.3.20 Changes for page: ProjectRequirements

- (11/01/05 15:04:08 by smithswe): *standardized page layout format*
- (10/25/05 16:05:05 by smithswe): *Removed "See Also."*
- (10/20/05 19:46:01 by arhedges): *No comment*
- (10/09/05 18:18:50 by gsymons): *Added header and footer.*
- (10/06/05 07:45:02 by arhedges): *No comment*

12.3.21 Changes for page: ProjectWatches

- (11/01/05 15:03:17 by smithswe): *standardized page layout format*
- (10/25/05 16:07:23 by smithswe): *Removed "See Also."*
- (10/17/05 20:47:03 by gsymons): *Change issues to watches*

12.3.22 Changes for page: ProjectVelocity

- (11/03/05 23:12:23 by gsymons): *fixed header and cut project glossary from it.*
- (10/25/05 19:37:58 by smithswe): *No comment*
- (10/17/05 21:46:02 by gsymons): *No comment*

12.3.23 Changes for page: ReportingModule

- (11/03/05 18:57:39 by gsymons): *No comment*
- (11/03/05 18:56:38 by gsymons): *No comment*
- (11/01/05 15:07:05 by smithswe): *standardized page layout format*
- (10/20/05 19:55:20 by arhedges): *No comment*
- (10/20/05 01:06:11 by gsymons): *Fixed spelling. Again.*
- (10/20/05 01:05:42 by gsymons): *Fixed spelling.*
- (10/12/05 18:38:24 by smithswe): *Defined the reporting module*

12.3.24 Changes for page: UserStories

- (11/03/05 23:12:57 by gsymons): *fixed header and cut project glossary from it.*
- (10/25/05 19:36:06 by smithswe): *No comment*
- (10/20/05 19:57:55 by arhedges): *No comment*
- (10/17/05 22:20:49 by gsymons): *No comment*

12.3.25 Changes for page: TheCustomer

- (11/01/05 14:55:38 by smithswe): *No comment*
- (10/25/05 16:13:30 by smithswe): *No comment*
- (10/25/05 1